TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Rekonfigurovatelný systém na FPGA obvodu

Reconfigurable system on an FPGA

Studijní program: P2612 Elektrotechnika a informatika

Studijní obor: 2612V045 Technická kybernetika

Školící pracoviště: Ústav informačních technologií a elektroniky
Fakulta mechatroniky, informatiky a mezioborových studií
Technická univerzita v Liberci
Studentská 2/1402, 461 17, Liberec

Autor: Ing. Tomáš Drahoňovský

Školitel: Prof. Ing. Ondřej Novák, CSc.

Prohlášení

Byl jsem seznámen s tím, že na mou disertační práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 - školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé disertační práce pro vnitřní potřebu TUL.

Užiji-li disertační práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Disertační práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací se školitelem mé disertační práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí vloženou do IS STAG.

Datum:

Podpis:

Abstrakt

Práce se zabývá tvorbou metodiky návrhu rekonfigurovatelného systému na FPGA obvodu. Tato metodika využívá pokročilých technik založených na částečné dynamické rekonfiguraci za účelem optimalizace rekonfigurovatelných systémů z hlediska flexibility, vyžadované paměti, času potřebného pro implementaci návrhu a množství logických zdrojů FPGA obvodu nezbytného pro vytvoření rekonfigurovatelného systému.

V textu jsou představeny základní pojmy z oblastí struktury a konfigurace FPGA obvodů, dále pak základní vlastnosti částečné rekonfigurace, relokace částečných konfiguračních souborů, vyčítání konfigurační paměti FPGA a zapisování dat do interních registrů obvodu.

Jádro práce představuje metodiku návrhu rekonfigurovatelného systému s využitím výše zmíněných technik. Dílčí části této práce jsou ověřeny na různých experimentech. V závěru jsou shrnuty výsledky jednotlivých přístupů a diskutovány přínosy použitých technik.

Klíčová slova

FPGA, částečná dynamická rekonfigurace, vyčítání konfigurační paměti, relokace hardwarových úloh, optimalizace návrhu

Abstract

This work is focused on a methodology of the reconfigurable system design implemented on an FPGA. This methodology uses advanced techniques based on a partial dynamic reconfiguration in order to optimize a reconfigurable system in terms of system's flexibility, memory requirements, implementation time requirements and logic sources consumption.

The text describes basics of the FPGA structure and important features of the dynamic partial reconfiguration, partial bitstream relocation, FPGA's configuration memory readback and FPGA's internal registers states restoration techniques.

The main part of the work presents a design methodology of the reconfigurable system where all mentioned techniques are supported. Individual parts of this work were verified on several applications with different sizes. Conclusion summarizes the results of the different approaches and discussed the benefits of the involved techniques.

Key worlds

FPGA, partial dynamic reconfiguration, configuration memory readback, hardware task relocation, design optimization

Poděkování:

Na tomto místě bych rád poděkoval své rodině a nejbližším za podporu během doktorského studia a při tvorbě této práce. Dále bych rád poděkoval Kamile Klímové za její podporu a jazykovou korekturu této práce. Zároveň bych chtěl poděkovat svému školiteli prof. Ing. Ondřeji Novákovi, CSc., a kolegovi Ing. Martinu Rozkovcovi, PhD., za jejich cenné rady a zkušenosti.

Vytvoření této práce bylo finančně podpořeno v rámci projektu SGS na Technické univerzitě v Liberci.

Obsah

Seznam obrázků:

Seznam tabulek

Použité zkratky:

BitGen	Bitstream Generator
BRAM	Block RAM
CLB	Configurable Logic Block
CLK	CLocK
CMT	Clock Management Tile
CRC	Cyclic Redundancy Check
DIRT	DIrect Routing consTraints
DMA	Direct Memory Access
DP	Dynamic Part
DPR	Dynamic Partial Reconfiguration
DRC	Design Rule Check
DSP	Digital Signal Processing
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GSR	Global Set/Reset
GUI	Graphical User Interface
HDL	Hardware Description Language
IO	Input Output
JTAG	Joint Test Action Group
LUT	Look-Up-Table
NCD	Native Circuit Description
NGD	Native Generic Database
OTERA	Online Test strategies for Reliable Reconfigurable Architectures
PAR	Place And Route
PB	Partial Bitstream
PB	Programmable Block
PBR	Partial Bitstream Relocation
PCF	Physical Constraints File
PLB	Processor Local Bus
PRET	PRE-configuration online Test
PROT	Post-Reconfigurable Online Test

RAM	Random Access Memory
RISC	Reduced Instruction Set Computing
RM	Reconfigurable Module
RP	Reconfigurable Partition
SP	Static Part
SRAM	Static RAM
TMR	Triple Modular Redundancy
UART	Universal Asynchronous Receiver/Transmitter
UCF	User Constraints File
VHDL	Very High speed integrated circuits Description Language
VLIW	Very Long Instruction Word
XDL	Xilinx Design Language
XST	Xilinx Synthesis Tool

Slovník použitých cizích výrazů

bitstream – konfigurační soubor
bus macro – oddělovací makro
capture – uložení dat z interních registrů
configuration memory readback – vyčítání konfigurační paměti
netlist – popis propojení
proxy logic – oddělovací element
rawbitstream – konfigurační soubor v ASCII formátu
readback capture – vyčítání hodnot interních registrů
readback verify – vyčítání za účelem kontroly získaných dat
restore – zápis dat do interních registrů

Úvod

Složité číslicové systémy jsou v dnešní době nedílnou součástí celé škály moderních zařízení od spotřební elektroniky přes medicínské přístroje, zařízení pro testování a měření až po aplikace pro vojenskou a leteckou techniku. Při vývoji a výrobě těchto zařízení jsou stále častěji využívána programovatelná hradlová pole FPGA (Field Programmable Gate Array) [61].

Díky možnosti konfigurace obvodu kdykoli během jeho životního cyklu FPGA vykazují velkou flexibilitu. Paralelní zpracování dat a možnost vytvoření hardwarového modulu šitého přímo na míru dané aplikaci zase zaručují vysoký výkon těchto obvodů. Avšak ve chvíli, kdy konkrétní návrh do FPGA umístíme, cílové využití se značně zúží. To je způsobeno tím, že pro každou specifickou úlohu je třeba specifická komponenta a počet takovýchto komponent je omezen velikostí použitého obvodu. Tento fakt je možné minimalizovat, pokud se daný systém navrhne jako rekonfigurovatelný. Díky částečné rekonfiguraci je FPGA obvod schopen provádět funkce, které by v něm nemohly být implementovány současně z důvodu nedostatku logických zdrojů obvodu, a zároveň je možné měnit funkce částí systému bez toho, aby byl jeho zbytek nějak ovlivněn.

Rekonfigurovatelný systém představuje kompromis mezi systémem vytvořeným přímo pro danou aplikaci (aplikačně specifickým), který se vykazuje vysokým výkonem, a univerzálním FPGA systémem (sestavou obsahující hardwarové vybavení pro více typů aplikací) vyznačujícím se vysokou flexibilitou.

Motivací této práce je ověření možností využití pokročilých technik rozšiřujících částečnou dynamickou rekonfiguraci pro optimalizaci návrhu rekonfigurovatelných systémů na FPGA obvodu a zhodnocení možných přínosů využití vytvořené návrhové metodiky.

V první kapitole je vysvětlena struktura FPGA obvodů, možnosti jejich konfigurace a postup implementace návrhu. Kapitola dvě popisuje částečnou dynamickou rekonfiguraci a v kapitole tři jsou představeny cíle této disertační práce. Čtvrtá kapitola se zabývá vlastní metodikou návrhu rekonfigurovatelného systému a implementací jednotlivých částí. Pátá kapitola je věnována experimentům. V závěru jsou shrnuty dosažené výsledky a diskutovány přínosy této práce.

1 Obvody FPGA

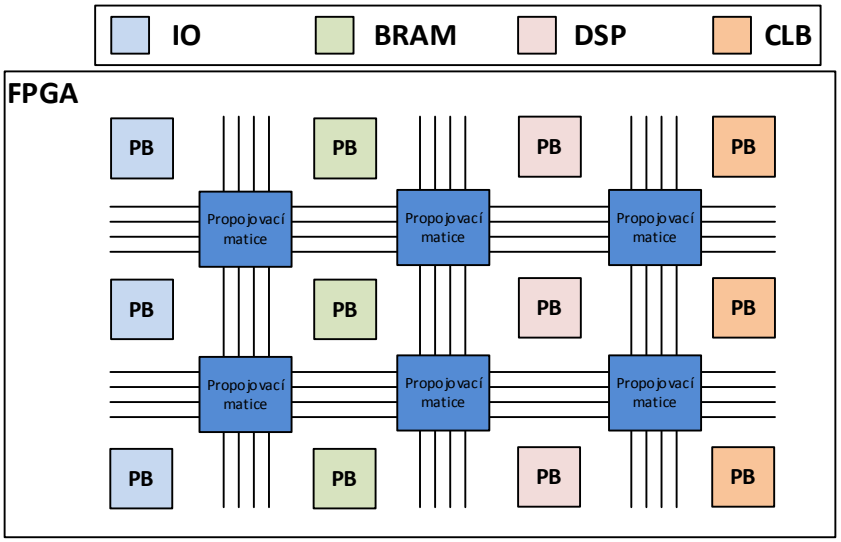
FPGA jsou programovatelné polovodičové součástky určené pro zpracování digitálních signálů s možností konfigurace kdykoliv ve svém životním cyklu. Použití FPGA přináší zkrácení času potřebného pro vývoj aplikace, vysokou flexibilitu a vysoký výkon [65].

Vzhledem k tomu, že výrobců těchto obvodů je celá řada (např. Xilinx, Altera, Actel atd.), není možné obecně popsat vnitřní strukturu všech FPGA, ačkoliv určitá podobnost mezi jednotlivými typy existuje. Z tohoto důvodu se v dalším textu zaměříme pouze na obvody společnosti Xilinx.

1.1 Struktura FPGA obvodů

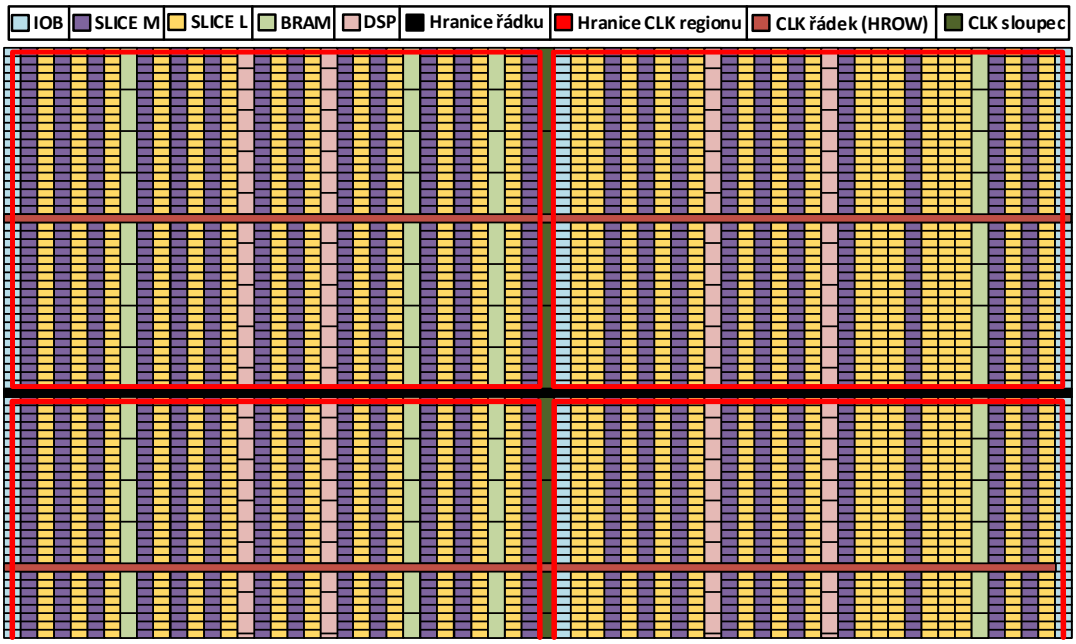
Následující popis se bude týkat obvodů z rodiny Virtex (konkrétně Virtex 6) od společnosti Xilinx. FPGA obvody rodiny Virtex se vyznačují vysokým výkonem s velkou hustotou vnitřní logiky (vysoký stupeň integrace). Kromě programovatelné logiky mají tyto obvody vestavěny hardwarové bloky pro implementaci složitých systémových funkcí, jako jsou násobičky, DSP (Digital Signal Processing) bloky, paměti RAM a procesorová jádra.

Virtex 6 je obvod vyráběný pomocí 40nm technologie. Je tvořen maticově uspořádanou strukturou různých programovatelných bloků (PB) vzájemně spojených nastavitelnou propojovací maticí. Toto je blokově naznačeno na Obr. 1.



Obr. 1: Principiální blokové schéma FPGA obvodu

Vnitřní struktura je uspořádaná do řádků a sloupců. Jednotlivé řádky jsou ještě rozděleny na hodinové (tzv. CLK) regiony. CLK regiony na levé a pravé straně obvodu jsou od sebe odděleny hodinovým (CLK) sloupcem, který prochází vertikálně středem obvodu. V tomto sloupci jsou umístěny globální a lokální hodinové přepínače řízené pomocí CMT (Clock Management Tile). Tyto přepínače slouží pro rozvod hodinového signálu do jednotlivých CLK regionů. Každý sloupec obsahuje vždy stejný typ logiky, ale jednotlivé sloupce jsou rozmístěny heterogenně po obvodu. To znamená, že sloupce obsahující jednotlivé logické zdroje se nestřídají pravidelně. Příklad rozmístění logických zdrojů obvodu Virtex 6 je naznačen na Obr. 2 [55].

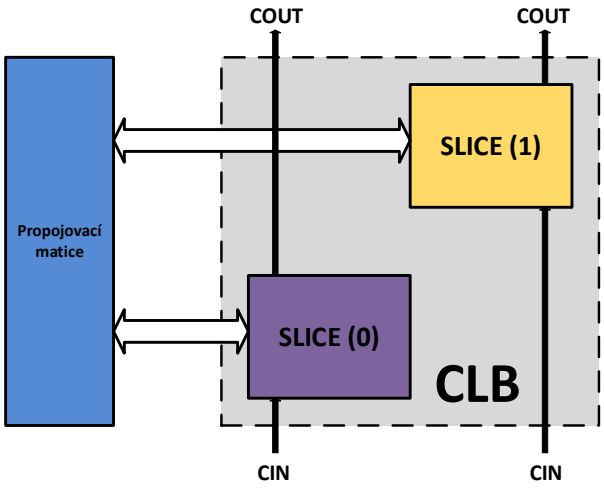


Obr. 2: Rozdělení vnitřní struktury FPGA obvodu

1.1.1 Základní logické prvky FPGA obvodů

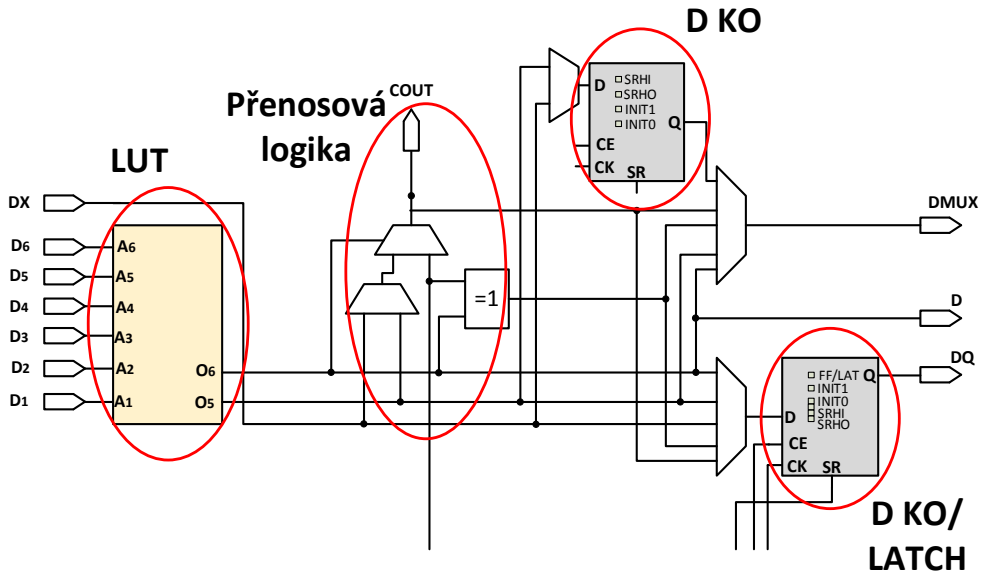
1.1.1.1 CLB bloky

Hlavním stavebním prvkem FPGA Virtex jsou tzv. CLB (Configurable Logic Block) bloky. CLB jsou složené ze dvou nižších funkčních celků (podbloků) nazývaných SLICE. Blokové uspořádání CLB je naznačeno na Obr. 3.



Obr. 3: Struktura konfigurovatelného logického bloku

U obvodů Virtex 6 rozeznáváme dva typy SLICE elementů – SLICE-L a SLICE-M. Každý SLICE je tvořen čtveřicí šestivstupových LUT (Look-Up-Table) elementů, výstupními multiplexory, osmicí paměťových prvků (D klopné obvody/LATCH) a přenosovou (carry) logikou. Přenosová logika se používá pro implementaci rychlých aritmetických operací a pro kaskádní propojení více LUT elementů (implementace velkých logických funkcí). LUT element může pracovat ve dvou různých režimech v závislosti na tom, ve kterém typu SLICE se nachází. V SLICE-L (Logic) je možné používat LUT elementy jako generátory libovolné logické funkce šesti vstupních proměnných. V SLICE-M (Memory) mohou LUT elementy kromě generátoru funkcí pracovat ještě jako 64bitová RAM paměť (tzv. distribuovaná RAM) nebo jako 32bitový posuvný registr. Na Obr. 4 je naznačená vnitřní struktura SLICE-L [65], [64].



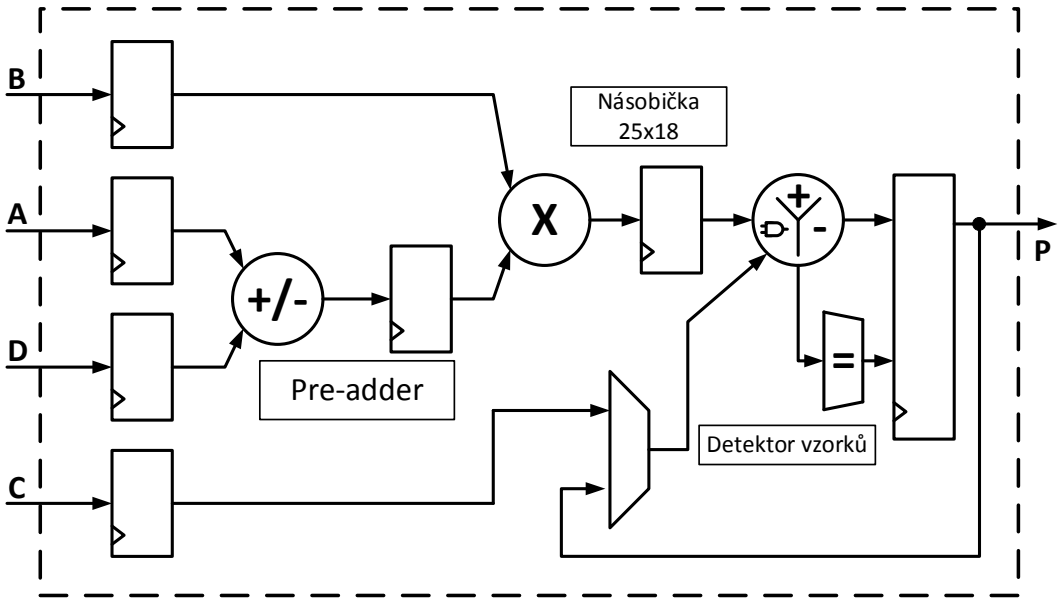
Obr. 4: Příklad vnitřního zapojení ¼ SLICE-L

1.1.1.2 DSP bloky

Vzhledem k možnosti implementace vlastních (libovolných) plně paralelních algoritmů jsou FPGA obvody velice efektivní při využití pro DSP (Digital Signal Processing) aplikace. Tyto aplikace ve velkém počtu využívají mnoho binárních násobiček, akumulátorů a dalších prvků, které je možné v FPGA snadno realizovat s využitím specializovaných DSP bloků. Obvody Virtex 6 obsahují velké množství těchto speciálních bloků, které umožňují kombinaci vysoké rychlosti zpracování a malých nároků na zabranou plochu obvodu [56]. Základní blokové schéma DSP bloku můžeme vidět na Obr. 5. DPS bloky nabízejí např.:

- binární násobičku 25 x 18 bitů pro čísla ve formátu dvojkového doplňku,
- 48bitový akumulátor,
- předzpracování (Pre-adder) některých vstupních signálů umožňující snížení napájecích nároků obvodu.

Množství dostupných DSP je závislé na velikosti FPGA. U obvodů Virtex 6 máme k dispozici od 288 bloků u nejmenších obvodů (XC6VLX75T) až po 2016 DSP bloků u obvodů největších (XC6VSX475T) [54].



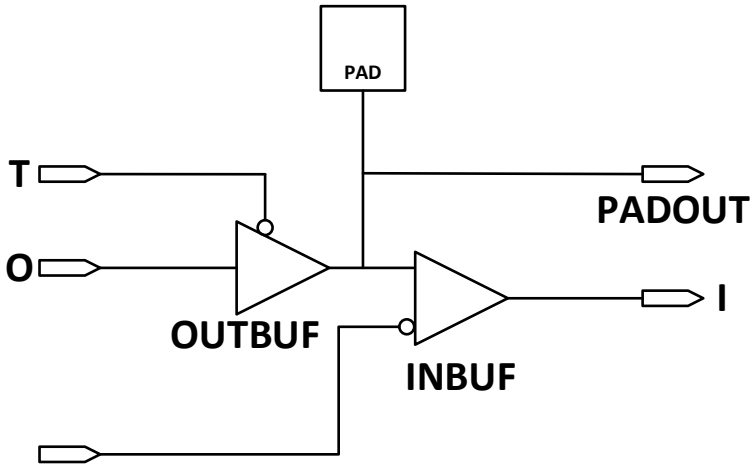
Obr. 5: Blokové schéma vnitřního uspořádání DSP bloku

1.1.1.3 Blokové paměti RAM

Každý obvod Virtex 6 má k dispozici značné množství systémové blokové paměti RAM (BRAM). Velikost bloku paměti je 36 kilobitů. Tento blok může být nakonfigurován i jako dvě nezávislé osmnáctikilobitové RAM paměti. BRAM paměti jsou dvouportové (dva symetrické zcela nezávislé porty sdílející pouze uložená data) s možností nastavení různé datové šířky jednotlivých portů (datová šířka portu pro čtení může být odlišná od šířky portu pro zápis). Obsah BRAM paměti může být inicializován ještě před spuštěním obvodu, a to při konfiguraci obvodu (nahráním konfiguračního souboru) [57]. Množství paměti dostupné v daném obvodu je opět závislé na jeho velikosti a pohybuje se od 5616 kilobitů, tj. 156 bloků (pro obvod XC6VLX75T), až po 38304 kilobitů, tj. 1064 bloků (pro obvod XC6VSX475T) [54].

1.1.1.4 IO bloky

Všechny FPGA Virtex 6 nabízejí programovatelné vstupně výstupní IO (Input/Output) bloky podporující řadu standardních rozhraní (LVTTL, LVCMOS2 atd.) s možností nastavení úrovně a rychlosti přeběhu výstupního signálu. Každý IO blok může být nakonfigurován jako vstup, výstup nebo jako třístavový IO (tj. umožňuje využívat stavu vysoké impedance). Na Obr. 6 můžeme vidět zjednodušené schéma IO bloku a jeho připojení k vnitřní logice a k pinu obvodu [58].

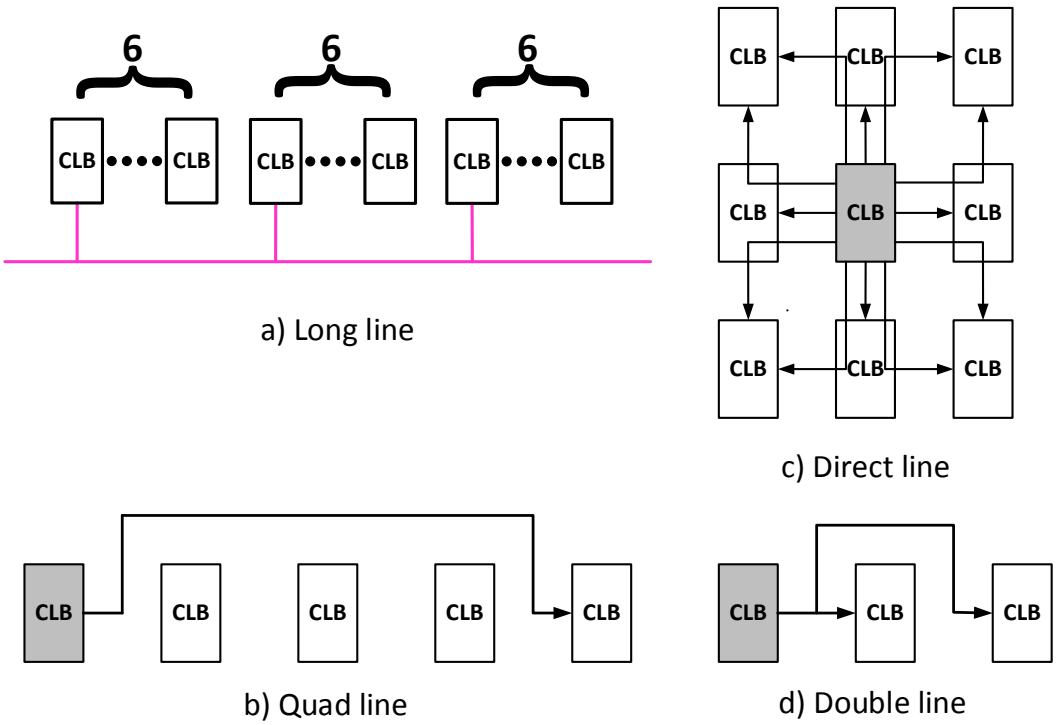


Obr. 6: Zjednodušené schéma IO bloku

1.1.2 Programovatelná propojovací síť

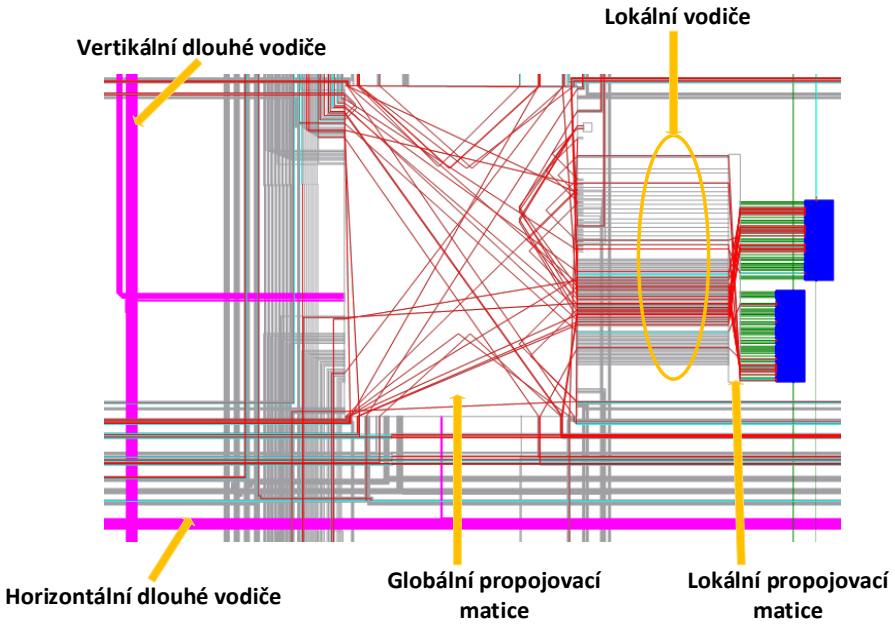
Stejně jako vnitřní strukturu FPGA obvodů různých výrobců, ani propojovací síť dokonce u jedné rodiny obvodů (Virtex) nelze zcela obecně popsat, proto se další popis bude týkat opět obvodů Virtex 6. Tyto obvody obsahují různé typy vodičů, které zajišťují propojení mezi programovatelnými logickými bloky. Tyto vodiče lze rozdělit na:

- dlouhé (vertikální a horizontální) vodiče (long lines) vhodné pro distribuci globálních signálů s malým zpožděním, tyto vodiče propojují každou šestou CLB (Obr. 7/a),
- tzv. čtvrtkové (vertikální a horizontální) vodiče (quad lines) vhodné pro vedení rychlých signálů s malým zpožděním umožňující efektivní propojení, protože propojují každé čtvrté CLB (Obr. 7/b),
- tzv. double (vertikální a horizontální) vodiče (double lines) spojující každou druhou CLB (Obr. 7/d),
- přímé vodiče (direct lines) sloužící pro propojení sousedních CLB (Obr. 7/c),
- lokální vodiče propojující jednotlivá CLB s lokální propojovací maticí,
- vodiče pro rozvod hodinového signálu.



Obr. 7: Propojovací vodiče u FPGA Virtex 6

Vodiče jsou pravidelně rozvedeny po celém obvodu a vzájemně propojeny pomocí lokálních a globálních programovatelných propojovacích matic symetricky rozmístěných po obvodu. Propojovací matice společně s některými typy vodičů jsou vyobrazeny na Obr. 8 [64].

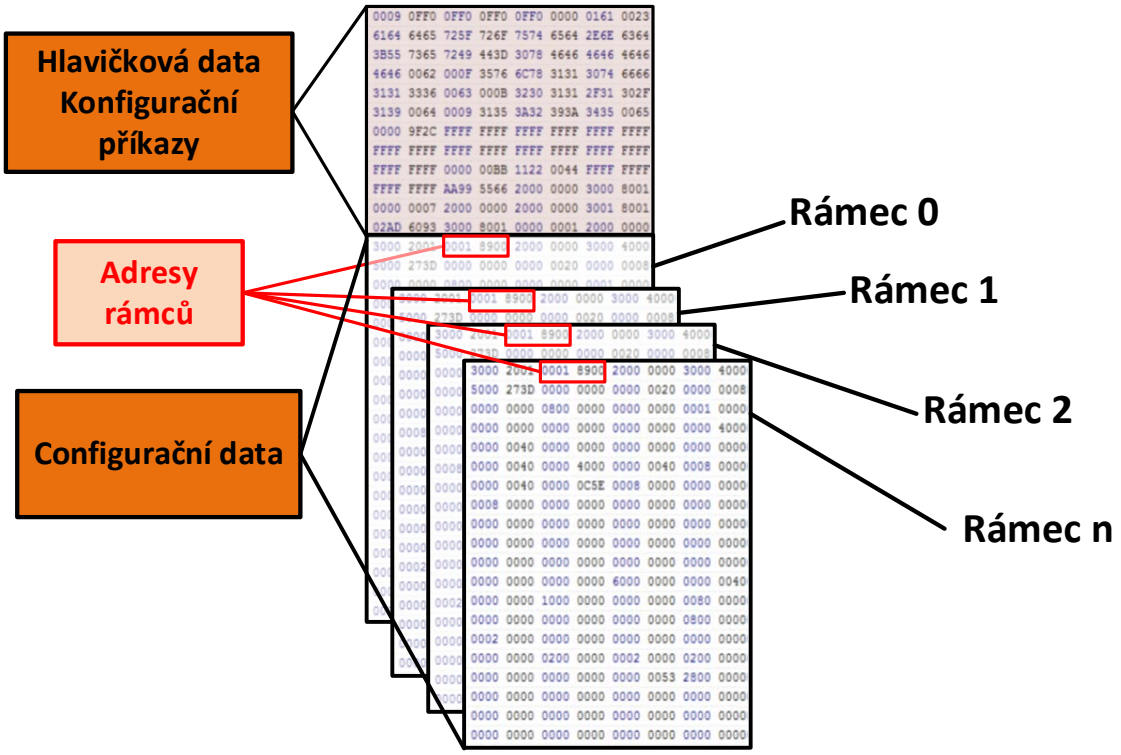


Obr. 8: Propojovací vodiče a programovatelná propojovací matice

1.2 Konfigurace FPGA obvodů

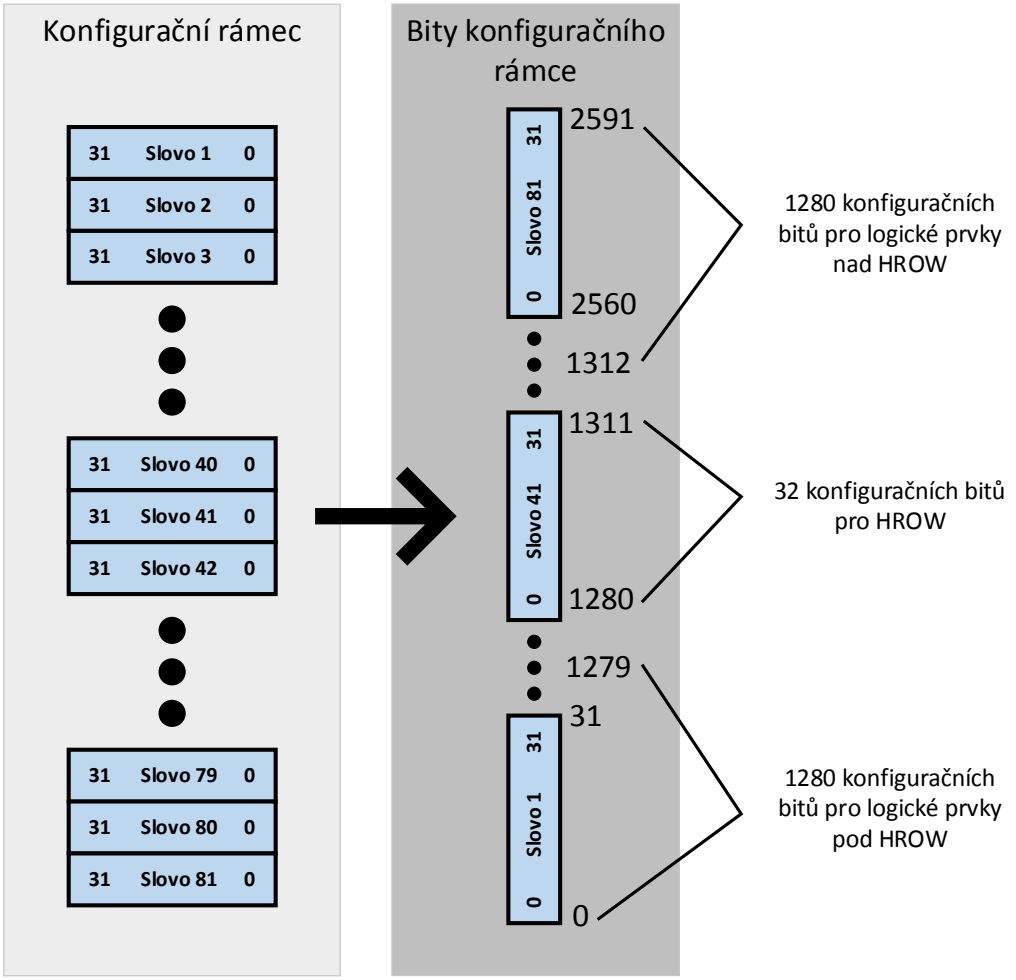
Funkce FPGA obvodu je dána obsahem konfigurační paměti typu SRAM. Jelikož se jedná o volatilní paměť, je třeba vždy po zapnutí napájení znovu nahrát data do této paměti. Konfigurační data jsou nahrávána v podobě konfiguračního souboru takzvaného bitstreamu. Jedná se o sekvenci 32bitových slov, obsahující jak konfigurační data pro dané FPGA, tak příkazy pro řídicí logiku obvodu. Rozlišujeme dva základní typy konfiguračních souborů – úplný a částečný. Úplný obsahuje konfigurační data pro celý FPGA obvod, zatímco částečný umožňuje nastavení pouze dané části obvodu. Faktem je, že částečný konfigurační soubor je v podstatě součástí toho úplného (při vytváření úplného konfiguračního souboru vždy musíme zahrnout data pro konfiguraci rekonfigurovatelných částí). Část konfiguračního souboru je vyobrazena na Obr. 9.

Více informací o základních konfiguračních příkazech, které jsou v konfiguračním souboru použity, lze nalézt v [65] a [53].



Obr. 9: Část konfiguračního souboru

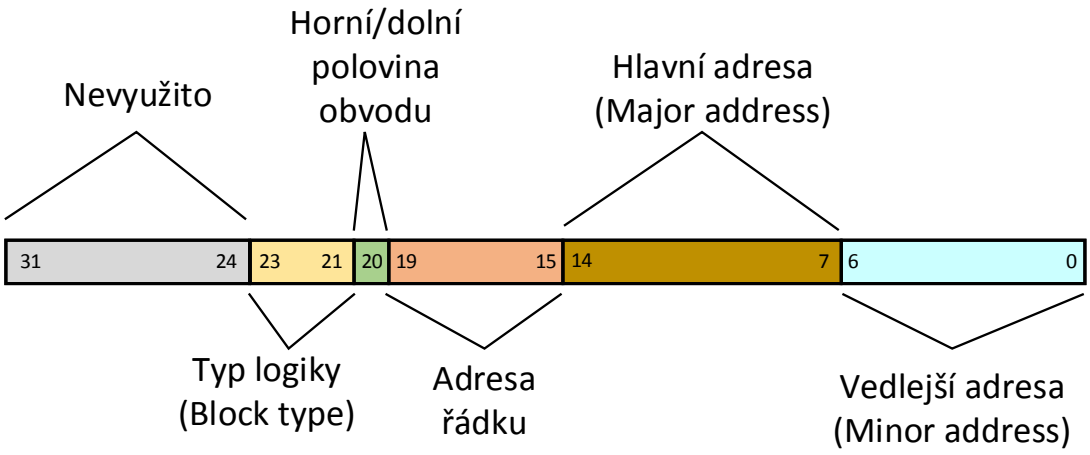
Konfigurační data jsou organizována do tzv. konfiguračních rámců (frames). Tento rámec je nejmenší adresovatelný segment konfigurační paměti, ke kterému lze přistupovat. Konfigurační rámec (pro obvody Virtex 6) si lze představit jako sloupec 2592 bitů (81 x 32 bitů) rozprostřených na celou výšku CLK řádku (viz Obr. 2). Jednotlivé sloupce v každém CLK řádku jsou tvořeny ze základních stavebních bloků (tj. např. 40 CLB, 40 IOB, osm BRAM, šestnáct DSP atd.), kde středem každého sloupce prochází tzv. hodinový řádek (HROW) obsahující CLK přepínače. Ze zmíněných 2592 bitů v každém sloupci prvních 1280 bitů určuje konfiguraci prvků v horní polovině sloupce (tj. nad HROW), dalších 1280 bitů zajišťuje nastavení prvků ve spodní polovině sloupce (tj. pod HROW). Zbylých 32 bitů je vyhrazeno (některé jsou nepoužité) pro konfiguraci samotného HROW. Toto rozložení platí pro všechny řádky v celém FPGA a je naznačeno na Obr. 10.



Obr. 10: Rozložení bitů konfiguračního rámce

Pozice každého rámce uvnitř obvodu je dána unikátní adresou (frame address). Pokud je prováděna konfigurace více na sebe navazujících rámců současně, může být (při běžném nastavení je) adresa rámce v konfiguračním souboru uvedena pouze jednou a během konfigurace je automaticky inkrementována. Pokud je vyžadováno (většinou za účelem odlaďování systému) mít v konfiguračním souboru zobrazeny adresy všech rámců. Pro tuto funkci je třeba v generátoru konfiguračních souborů (BitGen) nastavit parametr: *–g debugbitstream:yes*.

Adresa každého rámce má velikost 32 bitů a je rozdělena na pět částí. Struktura adresy rámce je vyobrazena na Obr. 11. Více podrobností o adresaci rámců je možné nalézt v [65] a [53]. V dalším textu jsou jednotlivé části stručně popsány.



Obr. 11: Rozdělení 32bitové adresy rámce na 5 částí

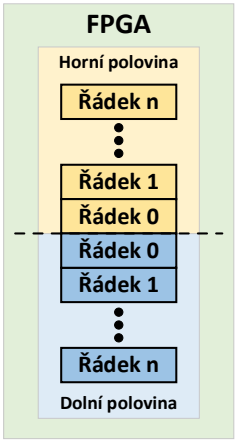
Typ logiky (Block type)

Tato část adresy určuje, který konkrétní typ logiky uvnitř obvodu je právě adresován. U obvodů Virtex 6 můžeme adresovat následující konfigurace:

- propojení a konfiguraci běžných rámců. Jedná se o konfiguraci standardních logických bloků (CLB, DSP, IOB atd.) a konfiguraci parametrů BRAM (např. šířka portu), nikoliv však obsah těchto pamětí,
- obsah BRAM pamětí. Obsah BRAM je samostatně konfigurován ze dvou důvodů. Tím prvním je jednoduchý fakt, že práce s konfiguračními rámci určujícími obsah paměti je odlišná od konfiguračních rámců běžné logiky. Druhým důvodem je to, že v případě, kdy obsah BRAM nepotřebujeme při konfiguraci FPGA nastavovat, můžeme tyto rámce snadno vynechat při vytváření konfiguračního souboru a tím notně zmenšit jeho velikost,
- propojení a konfiguraci speciálních (nestandardních) rámců. V každém sloupci se nachází jeden speciální rámec obsahující konfigurační bity využívané při částečné rekonfiguraci FPGA. Vzhledem k tomu, že částečná rekonfigurace je využívána pouze u malého procenta návrhů, jsou tyto rámce adresovány samostatně, aby bylo možné je snadno vynechat při vytváření konfiguračního souboru (obdobně jako v předchozím případě). Částečná rekonfigurace bude podrobně vysvětlena v následujících kapitolách.

Horní/dolní část obvodu a adresa řádku

Jak již bylo řečeno v předcházejících kapitolách, vnitřní struktura FPGA Virtex 6 je rozdělena do řádků a sloupců. Tento fakt se odráží i ve struktuře adresy rámce, kde jsou jednotlivé řádky adresovány od středu obvodu zvlášť v horní a dolní polovině (vždy od nuly). Toto rozdělení FPGA obvodu je naznačeno na Obr. 12.



Obr. 12: Rozdělení obvodu na poloviny pro účely adresování

Hlavní adresa (Major address)

Každý řádek v FPGA obsahuje stejný počet sloupců, které odpovídají jednotlivým logickým prvkům (CLB, DSP, IOB atd.). Hlavní adresa udává pozici těchto sloupců v obvodu s adresou nula začínající zleva. Existují dvě možné sekvence hlavní adresy na každý řádek, přičemž jedna je používána pro adresaci sloupců obsahujících standardní logické bloky a druhá sekvence provádí adresaci obsahu pamětí BRAM. To znamená, že BRAM mají dvě hlavní adresy – jedna slouží pro konfiguraci samotných pamětí, druhá umožňuje adresovat jejich obsah.

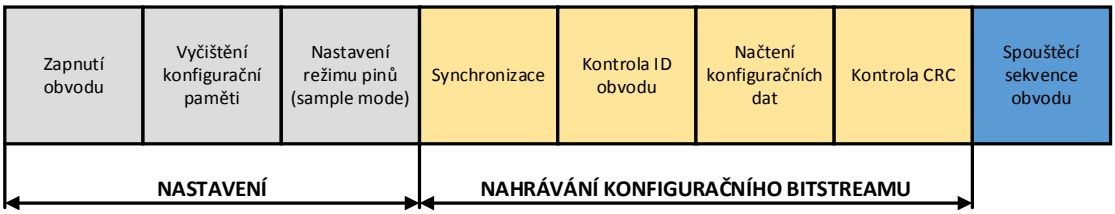
Vedlejší adresa (Minor address)

Každý sloupec v obvodu je nastavován určitým počtem konfiguračních rámců. Tento počet se odvíjí od toho, které logické prvky jsou v daném sloupci obsaženy (CLB, DSP atd.), a od toho, který typ logiky (Block type) je v daném sloupci adresován. Vedlejší adresa určuje, který konfigurační rámec v daném sloupci je právě adresován.

Samotná konfigurace těchto FPGA obvodů je prováděna nahráním konfiguračního souboru do vnitřní paměti obvodu. Tento soubor může být do FPGA stažen z paměti nebo může být nahrán s využitím externího zařízení (mikroprocesor, PC, tester atd.).

Obecně lze říci, že rozlišujeme dvě cesty, jak FPGA obvod konfigurovat. Prvním způsobem je sériová konfigurace – tato možnost se využívá za účelem minimalizace počtu potřebných pinů. Detailní popis je k nalezení v [65]. Druhou možností je použití paralelní konfigurace s datovou šířkou osm, šestnáct nebo 32 bitů. Paralelní způsob lze ještě rozdělit na několik módů podle typu zdroje (interní/externí) hodinového signálu na master a slave konfigurační mód.

Ačkoli jednotlivé způsoby se od sebe liší, konfigurační řetězec je možné obecně popsat. Tento řetězec je vyobrazen na Obr. 13. V dalším textu jsou stručně popsána základní konfigurační rozhraní.



Obr. 13: Jednotlivé části konfiguračního řetězce

SelectMAP

SelectMAP je obousměrné konfigurační rozhraní, to znamená, že s jeho pomocí lze jak nastavovat FPGA obvod, tak vyčítat konfigurační paměti (tato technika bude popsána v následujících kapitolách). Toto paralelní rozhraní používá datová slova o šířce osmi, šestnácti, nebo 32 bitů a jeho šířka je automaticky detekována. Rozhraní SelectMAP může být použito pro konfiguraci jednoho obvodu nebo více FPGA zároveň (využití v multi-FPGA systémech, do všech obvodů je nahrávána stejná konfigurace). Více podrobností o tomto rozhraní je k nalezení v [65], [53].

ICAP

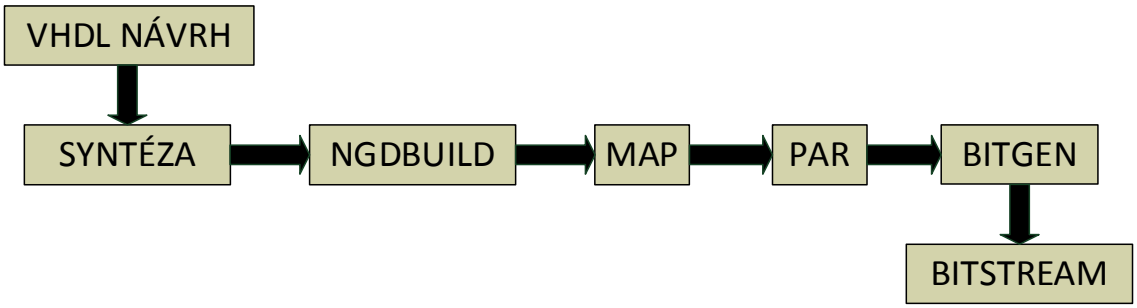
ICAP (Internal Configuration Acces Port) je interní konfigurační port, který je velice podobný rozhraní SelectMAP s tím rozdílem, že SelectMAP využívá obousměrného datového portu a rozhraní ICAP má oddělené vstupní a výstupní datové kanály. Přes rozhraní ICAP je možné přistupovat k uživatelským registrům, vyčítat konfigurační data nebo provádět částečnou rekonfiguraci FPGA obvodů.

JTAG

JTAG (Joint Test Action Group) je zjednodušený název sériového rozhraní, které je jinak definované jako IEEE 1149.1 Standard Test Access Port (TAP) and Boundary-Scan Architecture. Toto rozhraní bylo navrženo hlavně pro testování desek plošných spojů technikou Boundery-Scan. Je také využíváno pro konfiguraci a odlaďování integrovaných obvodů a vestavných (embedded) systémů. Jedná se o nejjednodušší, ale zároveň nejpomalejší způsob konfigurace FPGA obvodů.

1.3 Implementace návrhu do FPGA Xilinx

Pro získání konfiguračního souboru je nutné, aby návrh popsaný pomocí nějakého HDL (Hardware Description Language) jazyka prošel několika implementačními kroky. Nejprve je třeba návrh (většinou vytvořený v jazyce VHDL, Verilog, případně jako schéma) syntetizovat. Tím dojde k převodu informace o požadovaném chování (behaviorální popis) návrhu na strukturní netlist a optimalizaci návrhu pro dané zařízení. U obvodů Xilinx se syntéza provádí pomocí nástroje XST (Xilinx Synthesis Tool). Dalším krokem je překlad strukturního netlistu nástrojem zvaným NGDBuild. Ten vytvoří tzv. NGD (Native Generic Database) soubor, který obsahuje popis návrhu na úrovni obsažených logických elementů (hradla, LUT elementy, klopné obvody, RAM paměti atd.). Jednotlivé logické elementy jsou dále nástrojem MAP namapované a umístěné do konkrétních logických prvků (CLB, IOB, DSP atd.) obsažených na daném FPGA. Po namapování návrhu jsou tyto komponenty vzájemně propojeny nástrojem PAR (Place And Route). Ačkoliv název tohoto nástroje – Place and Route, napovídá, že k rozmísťování jednotlivých komponent dochází až v tomto kroku, není tomu tak. Komponenty jsou umístěny již při mapování a PAR provádí pouze propojení návrhu. Tento nástroj umožňuje provést znovu umístění (Re-Pleacing), pokud je to požadováno. Posledním krokem implementačního řetězce je vytvoření konfiguračního souboru nástrojem BitGen (BITstream GENerator). Blokové schéma implementačního řetězce s využitím návrhových prostředků firmy Xilinx je vyobrazeno na Obr. 14.



Obr. 14: Implementační řetězec obvodů Xilinx

Jednotlivé nástroje v řetězci umožňují celou řadu uživatelských nastavení, podrobné informace o všech možnostech implementace jsou uvedeny v [19]. Každý z těchto nástrojů po skončení své činnosti vytváří podrobný report s informacemi o průběhu dané části implementace. Vedle samotného zdrojového souboru s popisem celého návrhu (např. VHDL), který lze považovat za vstupní soubor všech nástrojů (ač je v průběhu samotné implementace tento soubor převáděn do různých formátů), existují další vstupní soubory, kterými lze ovlivnit průběh implementace.

Vstupními soubory používanými pro zadávání podmínek a omezení implementačních kroků jsou soubory typu UCF (User Constraints File) a PCF (Physical Constraints File). UCF soubor je jedním ze vstupních souborů nástroje NGDBUILD a umožňuje vkládat uživatelská omezení vztahující se jak k časování prvků v návrhu, tak i k jejich umístění v obvodu. PCF soubor je vstupním souborem nástroje PAR a obsahuje konkrétní fyzická omezení. Tento soubor je zpravidla vytvářen automaticky jako výstup nástroje MAP.

2 Částečná dynamická rekonfigurace FPGA

Částečná dynamická rekonfigurace (DPR – Dynamic Partial Reconfiguration) je technika umožňující za běhu obvodu měnit některé jeho části, přičemž zbytek obvodu není touto změnou nijak ovlivněn [62].

S částečnou rekonfigurací FPGA obvodů se dnes můžeme setkat u celé řady vědeckých i praktických aplikací. Např. v [24] a [25] je popsaný vývoj a implementace dynamicky rekonfigurovatelného systému určeného pro zpracování otisků prstů (automatické ověření otisků prstů) v reálném čase (real time). DPR je zde využita pro změnu funkce obvodu při různých fázích zpracování otisků prstů (normalizace obrazu, filtrování atd.). Použití DPR značně zmenšuje velikost FPGA obvodu potřebného pro tuto aplikaci. V podobném duchu je navržena i aplikace popsaná v [37] – jedná se o dynamicky rekonfigurovatelný systém na FPGA obvodu určený pro sledování provozu na komunikační síti a případné odhalování provozních anomálií. Díky částečné rekonfiguraci je u tohoto systému možné třikrát snížit množství potřebné logiky oproti systému bez rekonfigurace. V [39] je popsány návrh a implementace rekonfigurovatelného FIR (Finite Impulse Response) filtru s možností dynamicky měnit parametry tohoto filtru.

RAMPSoC [23] je víceprocesorový systém s možností rekonfigurace jednotlivých koprocesorů, komunikační infrastruktury a dokonce i procesorů samotných. Celý systém je kontrolován speciálním operačním systémem.

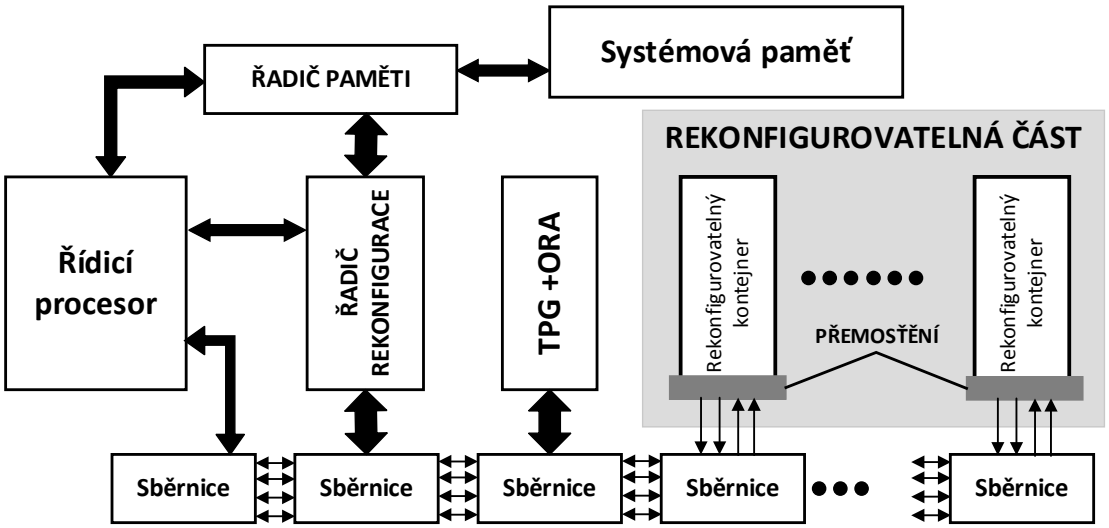
Využití DPR pro zlepšení vlastností síťových senzorů je popsáno v [26]. MFCA [44] je rekonfigurovatelný systém složený z více FPGA obvodů (cluster) využívající DPR uživatelských IP (Internal Peripheral). Rekonfigurace je obsluhována vestavěným procesorem PowerPC, který je řízen operačním systémem Linux. V [17] jsou obsaženy informace o platformě, která je založená na procesoru ARM9 s podporou rekonfigurace jednotlivých systémových komponent. V [12] autoři popisují systém s rekonfigurovatelným VLIW (Very Long Instruction Word) procesorem. Tato aplikace je založena na soft procesoru s možností dynamicky měnit počet jader a vláken procesoru (využití dvou jader s jedním vláknem, dvou jader se dvěma vlákny nebo jednoho jádra se čtyřmi vlákny). V případě dvoujádrového režimu je možné uvedení nevyužitých jader do nízkoodběrového módu. Uvedený systém je realizovaný na FPGA Virtex II Pro a Virtex 4.

S částečnou rekonfigurací se můžeme setkat i u celé řady aplikací určených pro řízení spolehlivosti FPGA systémů. V první řadě se jedná o systémy využívající částečné rekonfigurace pro testování jednotlivých částí obvodu.

Představitelem testovací techniky využívající DPR je např. roving STAR algoritmus [11]. FPGA obvod je rozdělen na stejně široké sloupce a řádky tak, že alespoň jeden sloupec a řádek zůstává neobsazen. Do neobsazeného prostoru je umístěn tzv. STAR (Self-Testing ARea). Oblast STAR je podrobena testu, zatímco zbytek obvodu, který není testováním ovlivněn, může vykonávat původní funkci. Test jiné části obvodu je realizován tak, že se STAR přesune do jiného sloupce obvodu. Při přesouvání STAR je nutné činnost obvodu pozastavit. Postupným přesouváním testovací oblasti je teoreticky možné otestovat celý obvod, aniž by byla narušena jeho funkce.

Částečnou rekonfiguraci využívá i OTERA [13] (Online Test strategies for reliable Reconfigurable Architectures). V tomto testovacím přístupu je kombinován strukturní online test určité části obvodu před její konfigurací (PRET – PRE-configuration online Test) s funkčním testem této části obvodu po její konfiguraci (PROT – Post-Reconfigurable Online Test), který ověří, zda rekonfigurace daného modulu proběhla v pořádku.

Architektura tohoto systému je rozdělena na statickou a rekonfigurovatelnou část. Statická část systému obsahuje řídicí procesor, řadič rekonfigurace, testovací vybavení a další statické komponenty. Rekonfigurovatelná část je složena z libovolného (omezeno velikostí obvodu) počtu „kontejnerů“ (oblastí). Tyto kontejnery mají statickou velikost, každý je umístěn na své statické pozici a každý je možné izolovat od zbytku systému za účelem provedení testu. Všechny kontejnery jsou propojeny jak mezi sebou, tak s testovacím vybavením. Blokové schéma systému OTERA je vyobrazeno na Obr. 15.



Obr. 15: Blokové naznačení rekonfigurovatelného systému OTERA

Existuje ještě celá řada prací zaměřených na využití částečné rekonfigurace při testování FPGA obvodů. Tyto aplikace využívají externí prostředky pro rekonfiguraci [49] nebo bývá využito procesorů vestavěných přímo do FPGA [41], [22], [21].

Další možností využití částečné rekonfigurace může být např. TMR (Triple Modular Redundancy) systém s možností opravy po výskytu poruchy v některém z modulů [16].

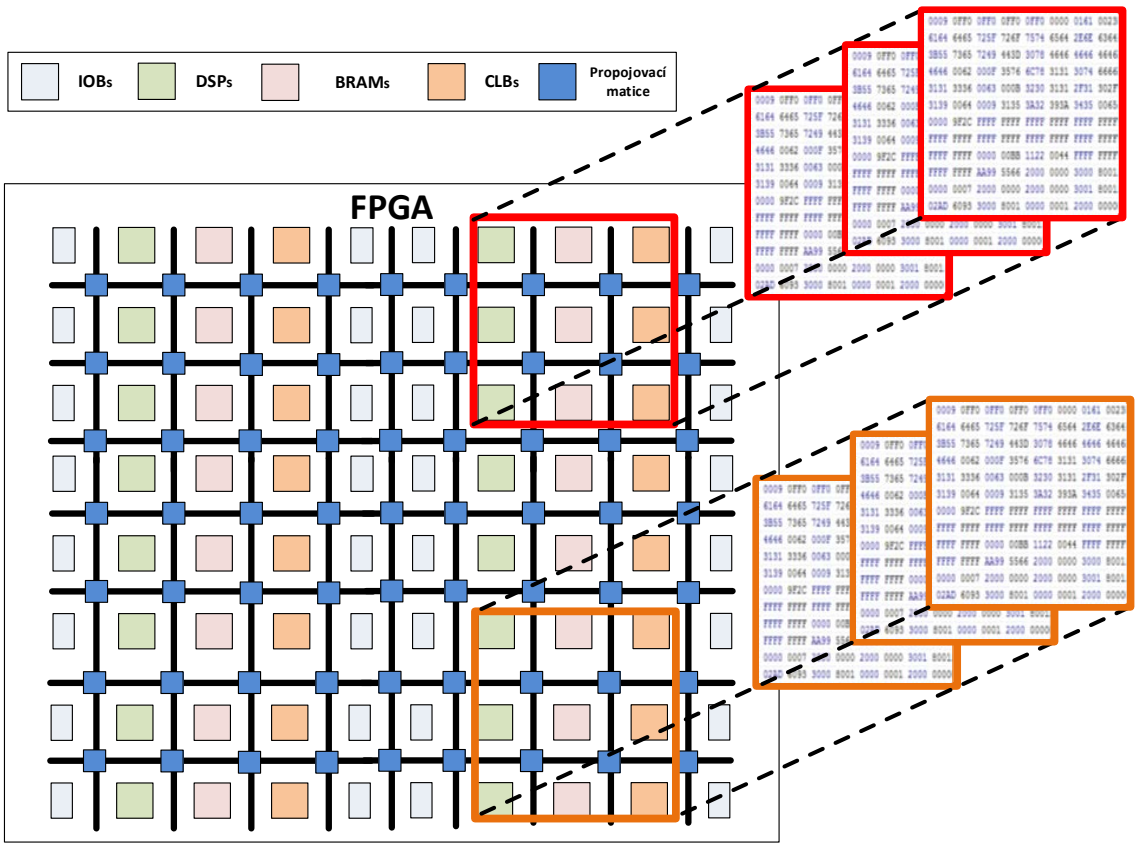
Rekonfigurací poruchového modulu sice dojde k odstranění poruchy, ale před jeho znovuzapojením do systému TMR je třeba tento modul synchronizovat se zbývajícími dvěma moduly. Existuje mnoho způsobů, jak tuto synchronizaci provést. Nejjednodušším řešením je po opravě aktivovat lokální reset celého TMR systému a tím nastavit počáteční stavy všech tří modulů. Tento přístup je využit např. v [47], kde systém po rekonfiguraci poruchového modulu vyčkává na nejvhodnější okamžik pro synchronizaci systému pomocí resetu. Do té doby pracuje systém pouze se dvěma moduly.

V případě, že moduly v TMR systému obsahují jednotky cyklicky opakující své stavy (FSM – Finite State Machine) [42], je možné mít vytvořený jakýsi kontrolní bod (check point), do kterého je modul po rekonfiguraci nastaven. Porucha na výstupu je maskována až do doby, kdy se zbylé funkční moduly dostanou do tohoto kontrolního bodu, po jehož dosažení je opravený modul zařazen zpět do TMR systému.

Další možností je fyzické propojení jednotlivých modulů pomocí datových vodičů nebo sběrnic [40]. V případě detekce poruchy je daný modul rekonfigurován, celý systém je pozastaven, vnitřní stavy z bezporuchových modulů jsou přeneseny do

opraveného modulu a celý systém je znovu spuštěn. Signálové propojení jednotlivých modulů nemusí být využíváno pouze k synchronizaci, ale i při běžném provozu. Tento případ je popsán v [48] a [34]. Prezentované systémy obsahují tři procesory MicroBlaze, v případě rekonfigurace jsou stavy funkčních procesorů uloženy do sdílené systémové paměti typu BRAM. Po skončení rekonfigurace je stejný stav nahrán do všech procesorů a systém je pak znovu spuštěn.

Technika DPR je v poslední době kromě obvodů firmy Xilinx podporována u některých FPGA jiných výrobců (např. Altera Startix V). Obvody firmy Xilinx však mají tuto techniku mnohem propracovanější, ať už po stránce podpory návrhu rekonfigurovatelného systému, nebo z hlediska samotné konfigurace jednotlivých částí obvodu. Z tohoto důvodu jsou v této práci využívány pouze obvody firmy Xilinx.



Obr. 16: Blokově naznačený princip částečné rekonfigurace

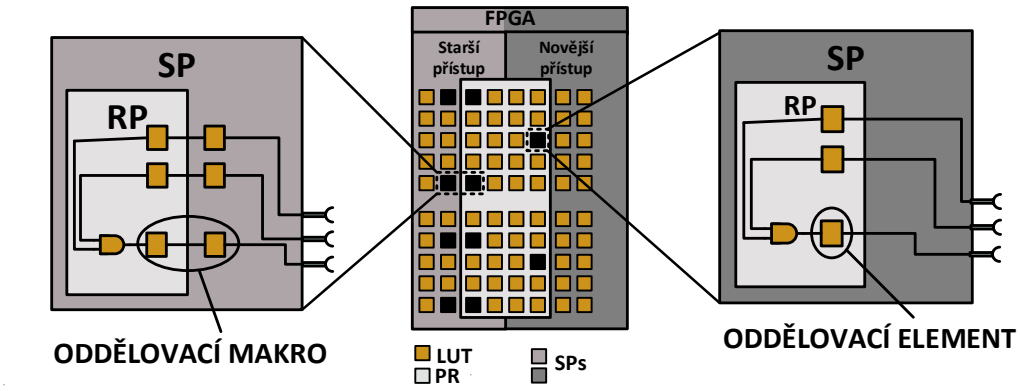
Princip DPR je blokově naznačen na Obr. 16. Vnitřní logika FPGA je rozdělena na statickou (SP – Static Part) a rekonfigurovatelnou (RP – Reconfigurable Part), někdy též nazývanou dynamickou (DP – Dynamic Part), část. V dynamické části je možné vytvářet rekonfigurovatelné oblasti (RP – Reconfigurable Partition), kam jsou umisťovány příslušné rekonfigurovatelné moduly (RM – Reconfigurable Module).

Funkce RP může být pozměněna nahráním jiného RM. Nahrání RM je provedeno stažením nového částečného konfiguračního souboru do konfigurační paměti obvodu (dojde k přepsání části konfigurační paměti FPGA obvodu). DPR v podstatě umožňuje časově multiplexovat části hardwaru na jednom FPGA, čímž lze dosáhnout:

- zmenšení velikosti zařízení FPGA potřebných pro provádění dané funkce,
- zvýšení flexibility při výběru algoritmů nebo protokolů, které má aplikace k dispozici,
- zavedení nových technik v provozuschopnosti návrhu,
- zvýšení odolnosti FPGA vůči chybám.

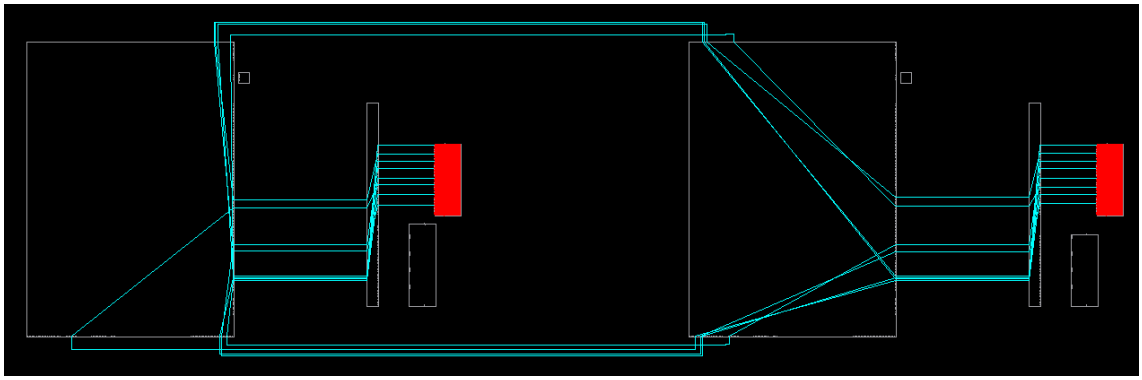
Kromě snížení velikosti, snížení nákladů a zvýšení výkonu částečná rekonfigurace umožňuje nové typy návrhů pro FPGA, které by bez částečné rekonfigurace nebylo možné realizovat [62]. Na druhou stranu použití DPR způsobuje větší složitost návrhu, zvyšuje čas potřebný pro implementaci návrhu, vnáší do systému větší nároky na paměť (paměť pro uložení částečných konfiguračních souborů) a způsobuje navýšení potřebných logických prvků (logic overhead).

Nárůst potřebné logiky je způsoben nutností oddělení statické a rekonfigurovatelné části. Signálové přemostění těchto částí může být realizováno dvěma způsoby. Prvním je signálové rozhraní nazývané bus macro [60] (obvody Virtex 4 a starší), které budeme označovat jako oddělovací makro. Jedná se o návrhářem vkládané rozhraní využívající dvou LUT elementů na každý jeden bit signálu (vodič) překračujícího hranici mezi SP a DP. Druhým způsobem je rozhraní nazývané proxy logika [62] (obvody Virtex 5 a novější), které budeme označovat jako oddělovací element. Oddělovací element je realizován jako jeden LUT element, automaticky vkládaný implementačními nástroji pro každý jeden bit signálu, který překračuje hranici mezi oběma regiony. Principiální schéma a základní rozdíl mezi oběma typy přemostění je vyobrazen na Obr. 17.



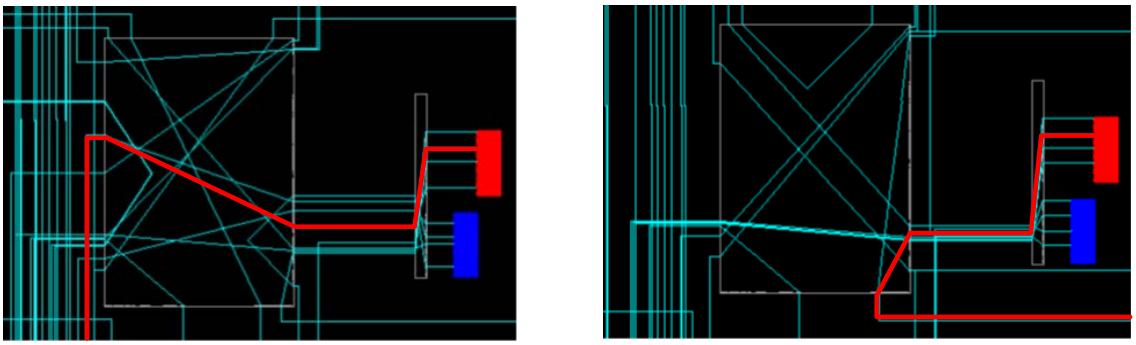
Obr. 17: Principiální naznačení realizace přemostění mezi SP a RP

Při použití staršího přístupu se jednotlivá oddělovací makra do návrhu vkládají jako přesně definované komponenty (využívá se soubor s příponou nmc) s pevně danou vnitřní strukturou (na fyzické úrovni). To znamená, že propojení LUT elementů tvořících oddělovací makro je vždy stejné pro všechny rekonfigurovatelné moduly v návrhu. Skutečnou podobu (výřez z programu Xilinx FPGA Editor) tohoto rozhraní můžeme vidět na Obr. 18.



Obr. 18: Skutečná podoba oddělovacího makra

Naproti tomu novější přístup s oddělovacími elementy sice vykazuje nižší nároky na dodatečné hardwarové vybavení (vždy jen jeden LUT element na jeden vodič), ale propojení mezi statickou a rekonfigurovatelnou částí je u jednotlivých modulů různé. To je naznačeno na Obr. 19. Můžeme zde vidět připojení stejného signálu pomocí oddělovacího elementu ve dvou identických rekonfigurovatelných modulech.



Obr. 19: Připojení oddělovacího elementu ve dvou identických RM

Nejmenší oblast, kterou lze samostatně rekonfigurovat se nazývá rekonfigurovatelný rámec. Nastavení každého takového rámce je dáno pomocí určitého počtu konfiguračních rámců (viz kapitola 1.2) – jejich počet, a tím i velikost konfiguračního souboru závisí na typu logiky obsažené v každém rekonfigurovatelném rámci. Obvody Virtex 6 využívají rekonfigurovatelný rámec o šíři jednoho CLB a výšce

40 CLB (tj. 160 LUT elementů), osm BRAM, nebo šestnáct DSP. Výška tohoto rámce odpovídá výšce hodinového regionu.

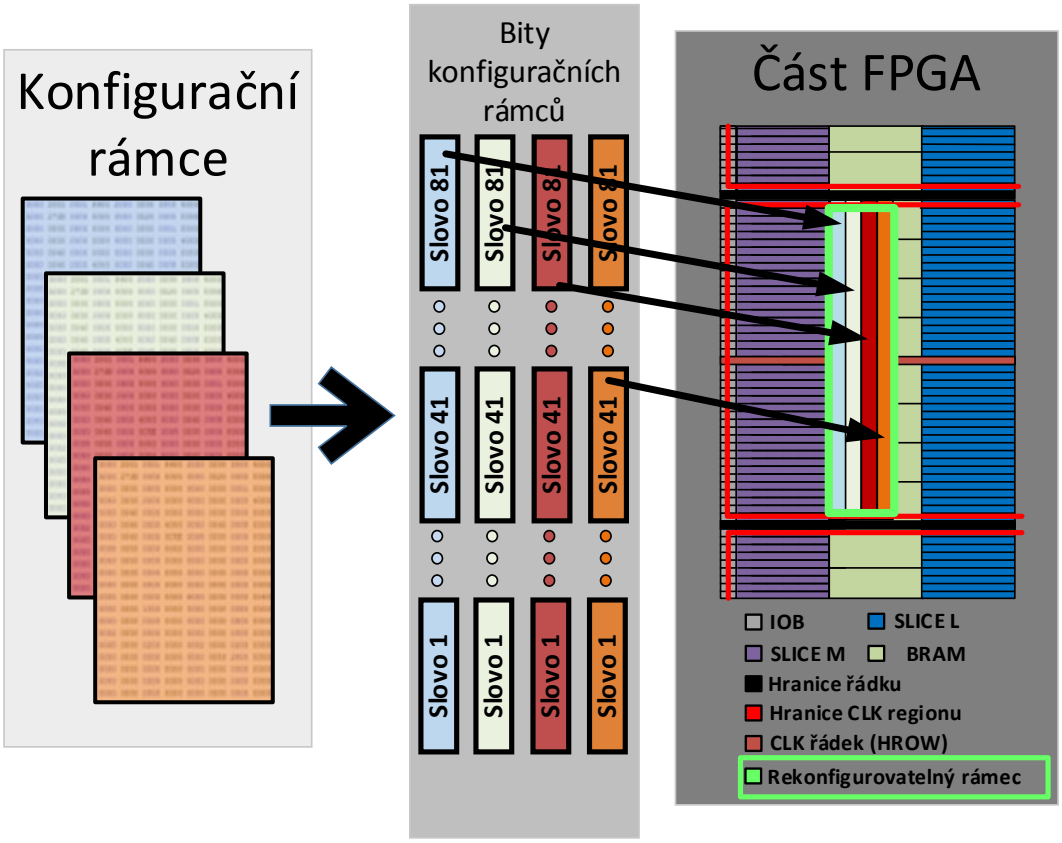
Pokud při vytváření návrhu definujeme menší rekonfigurovatelnou oblast, než je výška rámce, návrhový nástroj automaticky tuto oblast rozšíří na nejmenší povolenou velikost. Tento fakt má přímou souvislost s rozložením bitů v konfiguračním rámci, který nastavuje části logiky v celé výšce řádku (CLK regionu). Tato skutečnost je naznačena na Obr. 20. Tab. 1 zobrazuje přehled počtu konfiguračních rámců potřebných k nastavení jednoho rekonfigurovatelného rámce (jednoho sloupce o výšce CLK regionu) pro různé typy logiky.

První sloupec tabulky obsahuje typ logiky, která má být konfigurována. Druhý sloupec tabulky uvádí počet bitů potřebných pro nastavení jednoho rekonfigurovatelného rámce daného typu logiky. Třetí sloupec v tabulce říká, kolik konfiguračních rámců je zapotřebí pro nastavení jednoho rekonfigurovatelného rámce. Ve čtvrtém sloupci je nalezneme informaci o tom, kolik rekonfigurovatelných rámců zabírá jeden sloupec o výšce CLK regionu daného typu logiky.

Tab. 1: Přehled počtu konfiguračních rámců (pro obvody Virtex 6) potřebných pro konfiguraci rekonfigurovatelného rámce obsahujícího různé typy logiky

Typ logiky v daném sloupci	Počet konfiguračních bitů	Počet konfiguračních rámců	Počet rekonfigurovatelných rámců na jeden sloupec logiky
CLB	93319	36	1
BRAM konfigurace	75168	29	2*
BRAM obsah	334368	129	
DSP	72576	28	1
IOB	139968	54	1

*Pro konfiguraci jednoho sloupce pamětí BRAM je zapotřebí dvou rekonfigurovatelných rámců (jeden konfiguruje samotnou paměť, druhý obsah paměti).



Obr. 20: Souvislost mezi konfiguračním a rekonfigurovatelným rámcem

Obecně lze říci, že vytvoření rekonfigurovatelného návrhu je teoreticky možné na libovolném (Xilinx) FPGA. Ne všechny typy a rodiny obvodů podporují tuto techniku v rámci standardních návrhových prostředků. Existují návrhové techniky, např. [15] a [46], které umožňují rekonfigurovatelný návrh na libovolném obvodu. V dalším textu je stručně popsáno, jak vytvořit takovýto návrh s využitím běžně dostupných návrhových nástrojů (Xilinx ISE Design Tools verze 14.7). Konkrétně s využitím nástroje PlanAhead se zakoupenou licencí pro návrh systémů s podporou techniky DPR.

Při vytváření rekonfigurovatelného návrhu musíme provést několik základních úkonů:

- definice rekonfigurovatelných oblastí (RP),
- přiřazení rekonfigurovatelných modulů (RM) jednotlivým RP (počet RM přiřazených jednotlivým RP závisí na počtu funkcí, které má daná RP umožňovat),
- definice velikosti a tvaru jednotlivých RP (velikost a logické zdroje v jednotlivých RP musí být adekvátní k přiřazenému hardwarovému modulu),

- nastavení možností (strategie) implementace návrhu (nastavení funkcí a vlastností jednotlivých implementačních nástrojů),
- kontrola (DRC – Design Rule Check) správnosti návrhu (kontrola, zda je možné jednotlivé RM implementovat do vybraných RP),
- spuštění procesu implementace a generování konfiguračních souborů.

Celý výše (zjednodušeně) popsaný postup vytvoření rekonfigurovatelného návrhu může být proveden pomocí uživatelského rozhraní (GUI – Graphical User Interface) v návrhovém prostředí PlanAhead. Další možností je implementace s využitím příkazové řádky, případně je možná kombinace obou způsobů.

Ne všechny logické zdroje mohou být součástí RP, např. logika určená pro generování, úpravu a rozvod hodinových signálů musí vždy zůstat ve statické části návrhu. Stejně tak není možné rekonfigurovat všechny typy hardwarových modulů. Toto omezení se týká především modulů obsahujících signálové cesty vedoucí přímo ze vstupu na výstup (signál neprochází cestou na výstup přes žádnou logiku).

Čas potřebný pro provedení rekonfigurace hardwarového modulu (tj. čas potřebný pro nahrání nového konfiguračního souboru) je přímo úměrný velikosti tohoto souboru. Dále je závislý na typu a rychlosti konfiguračního rozhraní a na rychlosti, se kterou systém přistupuje k paměti, ve které je daný soubor uložen.

3 Cíle práce

Cílem této práce je zpracování koncepce návrhové metodiky dynamicky rekonfigurovatelného systému na FPGA obvodu. Podstata navržené metodiky spočívá v reflexi všech výhod částečné dynamické rekonfigurace (vysoký výkon, nízká spotřeba, vysoká flexibilita, vysoká provozuschopnost atd.) a maximální minimalizaci nevýhod této techniky (složitost návrhu, zvýšené nároky na paměť, požadavek na dodatečné hardwarové vybavení obvodu, doba nezbytná pro implementaci návrhu). Pro splnění cíle této práce se předpokládá splnění následujících dílčích cílů:

- analýza stávajících řešení rekonfigurovatelných systémů,
- seznámení s postupem návrhu rekonfigurovatelného systému,
- seznámení s pokročilými technikami založenými na částečné rekonfiguraci,
- návrh a vytvoření metody pro relokaci částečných konfiguračních souborů,
- návrh systémů využívajících zpětné vyčítaní konfigurační paměti FPGA obvodu,
- návrh systémů využívajících techniku zapsání dat z konfigurační paměti do interní logiky FPGA,
- implementace zmíněných technik do reálného FPGA systému,
- experimentální ověření funkce a předpokládaných vlastností jednotlivých technik.

Splněním cílů této práce umožníme vytvoření rekonfigurovatelného systému na FPGA obvodu s podporou částečné dynamické rekonfigurace, relokace rekonfigurovatelných modulů, zpětného vyčítání konfigurační paměti a zápisu dat z konfigurační paměti do interní logiky FPGA obvodu. Kombinace těchto technik umožní vytvoření komplexního, univerzálního a vysoce flexibilního systému na FPGA. Jedná se o systém podporující přístup s minimálními nároky na paměť pro uložení částečných konfiguračních souborů a s možností rekonfigurace jednotlivých částí tohoto systému včetně aktuálních hodnot interních registrů obvodu.

4 Navržený rekonfigurovatelný systém

S rekonfigurovatelnými systémy na FPGA obvodu se poslední dobou setkáváme stále častěji. Je to způsobeno především větší podporou této techniky v rámci standardních návrhových a implementačních nástrojů jednotlivých výrobců FPGA obvodů. Za nejpropracovanější lze považovat návrhové nástroje firmy Xilinx.

Způsobů využití částečné rekonfigurace existuje celá řada, od systémů určených pro zpracování velkého množství dat v reálném čase přes aplikace zaměřené na testování FPGA obvodů až po systémy s možností tolerance poruch.

Určit, u které aplikace je rekonfigurace přínosná a u které ne, lze jen těžko. Zahrnutí rekonfigurace do návrhu má ve všech případech za následek navýšení potřebné logiky (v závislosti na množství pinů rekonfigurovatelných oblastí) a navýšení potřebné paměti (v závislosti na množství funkcí jednotlivých oblastí).

Využitím techniky relokace hardwarových modulů lze množství potřebných konfiguračních souborů omezit na jeden soubor na každý modul implementovaný v obvodu. Této techniky lze snadno využít hlavně u systémů s podobnou nebo stejnou velikostí jednotlivých rekonfigurovatelných oblastí, jako je tomu např. v [13], kde by využití relokace vedlo k velké úspoře potřebné paměti.

Nejmenšího možného počtu potřebných částečných konfiguračních souborů dosáhneme kombinací techniky relokace hardwarových modulů s technikou zpětného vyčítání konfigurační paměti obvodu, kdy není zapotřebí konfigurační data ukládat do systémové paměti.

Dalším neduhem částečné rekonfigurace je navýšení potřebného hardwarového vybavení obvodu, které se projevuje hlavně u systémů, kde mají jednotlivé hardwarové komponenty velký počet vstupních a výstupních pinů, např. v [13]. Zde je ke každému modulu připojeno testovací vybavení. Nebo např. ve [40] a [47], kde je řešena synchronizace jednotlivých modulů jejich fyzickým propojením.

Použití datových sběrnic propojujících jednotlivé moduly zvyšuje náročnost propojení systému a navyšuje režii logických prvků potřebných pro částečnou rekonfiguraci (každý signál vyžaduje přemostění mezi statickou a dynamickou částí systému). Dále způsobuje zhoršení časování systému a navyšuje čas potřebný pro jeho implementaci (prodloužení doby potřebné pro propojení návrhu).

Použitím technik vyčítání konfigurační paměti a zápisu dat do interních registrů FPGA obvodu se lze oprostit od nutnosti tohoto fyzického propojení jednotlivých komponent.

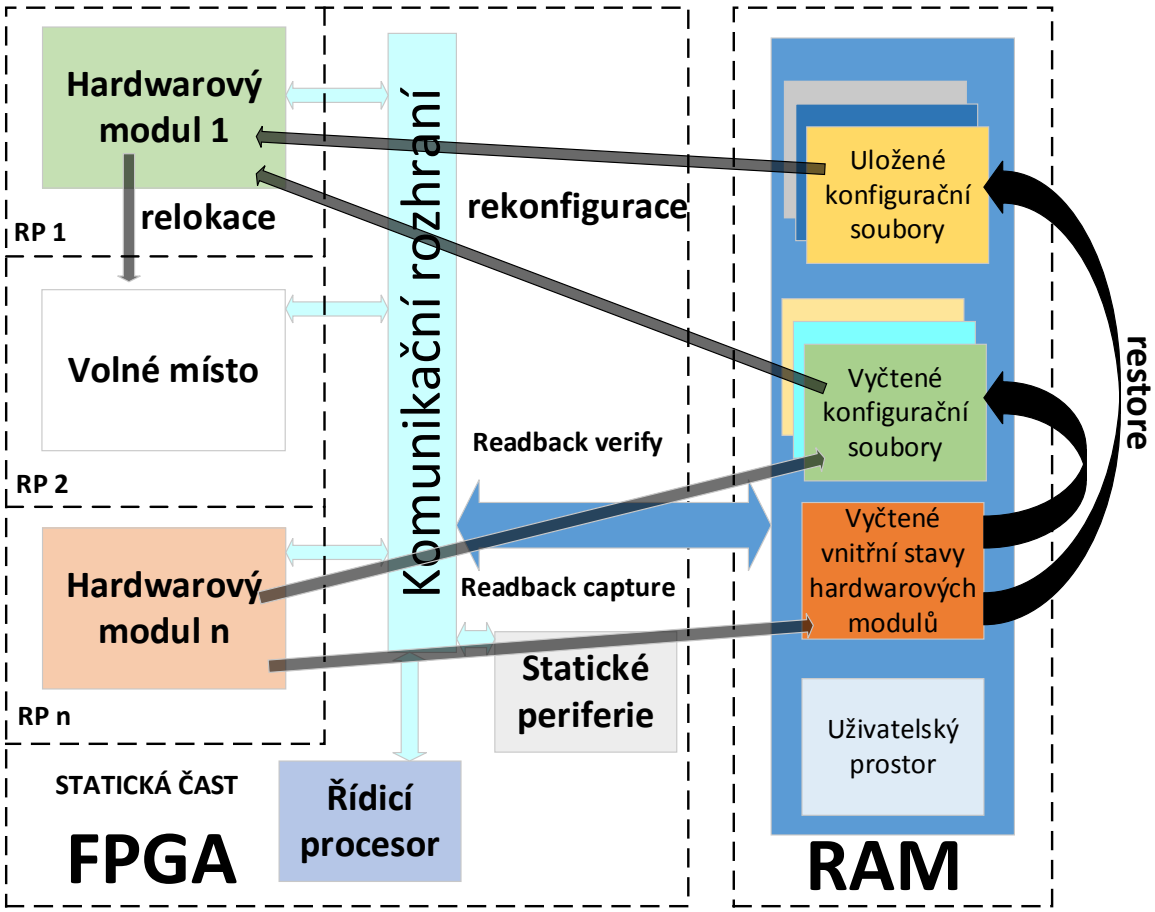
Prezentovaná metodika návrhu rekonfigurovatelného systému počítá s využitím všech zmíněných technik. Obr. 21 zobrazuje obecné schéma systému navrženého s využitím vytvořené metodiky. Návrh je rozdělen na jednu statickou a *n* rekonfigurovatelných částí (*n* – závisí na velikosti jednotlivých RP a na velikosti použitého obvodu). Statická část návrhu obsahuje řídicí procesor, který zajišťuje řízení celého systému a správu rekonfigurace. Procesor je přes komunikační rozhraní propojen s ostatními statickými komponentami (systémová paměť, ICAP, UART atd.) a s jednotlivými rekonfigurovatelnými moduly. Celý systém tvoří jakousi hvězdicovou strukturu.

Systém má k dispozici různé varianty hardwarových modulů, které jsou uloženy v paměti jako částečné konfigurační soubory nebo mohou být získány z konfigurační paměti obvodu technikou zpětného vyčítání. Tím lze dosáhnout snížení nároků na paměť potřebnou pro uložení těchto konfiguračních souborů.

Systém je navržen tak, aby využíval techniku DPR doplněnou o možnost relokace rekonfigurovatelných modulů, zpětného vyčítání konfigurační paměti a zápisu dat do interních registrů v obvodu.

Díky relokaci hardwarových modulů je možné využít jednotlivých částečných konfiguračních souborů na více místech v obvodu. S použitím zpětného vyčítání je možné relokovat komponenty získané přímo z konfigurační paměti obvodu. Zápis dat do interních registrů obvodu lze použít např. pro synchronizaci jednotlivých rekonfigurovatelných modulů, vkládání testovacích dat na libovolné místo v obvodu a relokaci jednotlivých komponent včetně jejich vnitřních stavů.

Využití zmíněných technik má za následek zvýšení flexibility a snížení paměťových nároků systému. Postup, jak systém podporující všechny zmíněné techniky vytvořit, i všechna úskalí s takovýmto návrhem spojená jsou popsány v následujících kapitolách.



Obr. 21: **Navržený rekonfigurovatelný systém (obecné blokové schéma)**

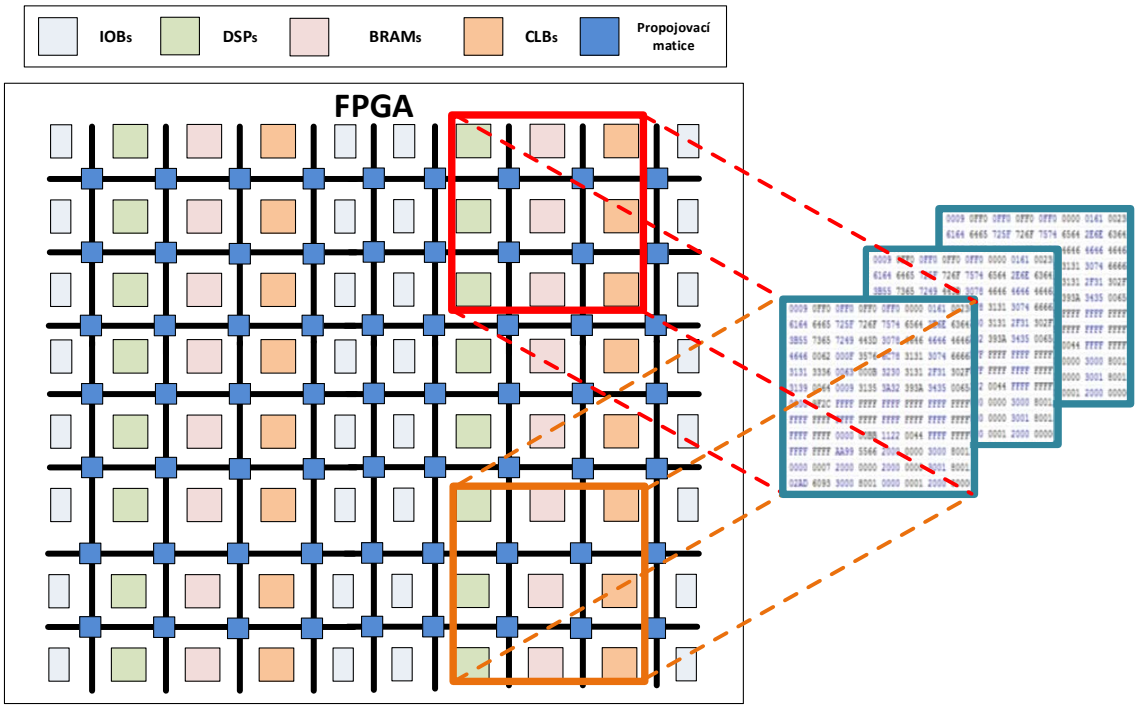
Postup změny funkce jednotlivých hardwarových modulů je následující:

1) zastavení činnosti daného modulu,
2) vyčtení a uložení vnitřních stavů rekonfigurovaného modulu (v případě potřeby),
3) rekonfigurace vybrané oblasti obvodu,
4) nastavení inicializace nového modulu (synchronizace),
5) spuštění nového modulu.

V průběhu rekonfigurace některé části systému může jeho zbytek pokračovat ve své dosavadní činnosti omezen pouze tím, že moduly, jejichž funkce je právě modifikována, nejsou k dispozici. Pokud je používána technika zpětného vyčítání konfigurační paměti, není zároveň možné provádět zápis do paměťových prvků, které jsou vyčítány.

4.1 Relokace částečných konfiguračních souborů

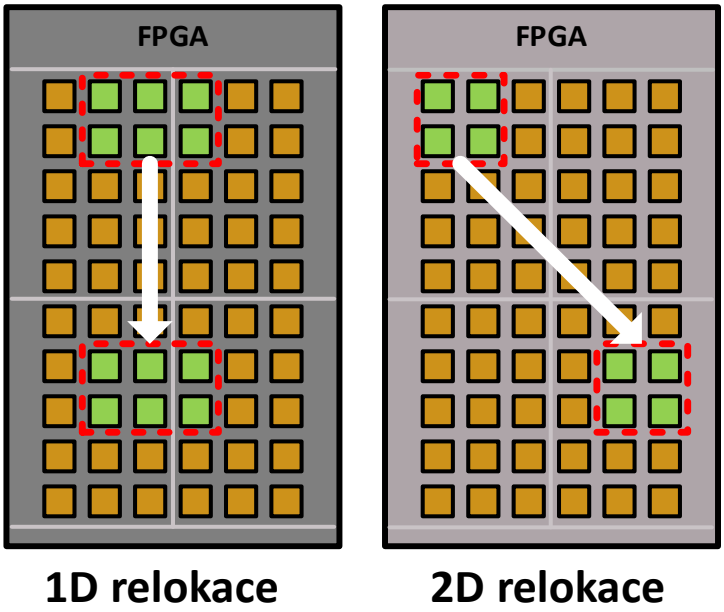
Relokace částečných konfiguračních souborů (PBR – Partial Bitstream Relocation) je technika založená na DPR. Tato technika umožňuje přesouvání jednotlivých hardwarových modulů umístěných původně v jedné části FPGA do jiné části obvodu. Zjednodušeně lze říci, že částečný konfigurační soubor vytvořený pro jednu konkrétní část obvodu je možné použít na více místech. Základní princip PBR techniky je naznačen na Obr. 22.



Obr. 22: Principiální blokové schéma PBR

Jak je řečeno výše (kapitola 2.1), vnitřní struktura většiny FPGA je uspořádána do sloupců a řádků. Podle toho, v jakém směru (sloupce/řádky) můžeme jednotlivé PB přesouvat (relokovat), rozlišujeme dva typy PBR techniky. Pokud je možné relokovat pouze ve sloupci (ve sloupci je možné relokovat téměř vždy – logické zdroje v jednotlivých sloupcích jsou stejné v celém obvodu), je tato relokace označována jako tzv. 1-D. Pokud je možné relokovat v obou směrech (ve sloupcích i v řádcích), mluvíme o tzv. 2-D relokaci [32].

Rozdíl mezi oběma typy je naznačen na Obr. 23. Typ relokace, který je možné v daném návrhu provádět, závisí na velikosti relokovaného PB, na stupni heterogenity a velikosti použitého FPGA. Stupněm heterogenity je myšleno, nakolik nerovnoměrně jsou jednotlivé logické prvky v daném obvodu rozmístěny.



Obr. 23: 1-D a 2-D relokace

4.1.1 Implementace systému s podporou PBR

Podpora techniky PBR je jedním ze základních principů využitých v této práci. Díky PBR lze dosáhnout zvýšení flexibility FPGA návrhu, snížení paměťových nároků systému v porovnání se systémy využívajícími standardní DPR a snížení časových nároků na implementaci návrhu, protože na vytvoření méně PB potřebujeme méně času. V neposlední řadě může PBR technika přispět ke zvýšení provozuschopnosti celého systému díky možnosti přesouvání hardwarových modulů z poruchové/poškozené do bezporuchové/nepoškozené části FPGA obvodu.

Postup pro vytvoření FPGA návrhu s podporou relokace částečných konfiguračních souborů není v zásadě příliš odlišný od návrhu systému, který je založený na standardní DPR. Během tohoto návrhu však musejí být splněny určité podmínky a jednotlivé oblasti, kam má být daný RM relokován, musejí splňovat jisté parametry.

Tyto návrhové podmínky lze shrnout v několika následujících bodech:

- stejná velikost všech RP,
- stejný počet, typ a rozmístění logických zdrojů v jednotlivých RP,
- žádné statické signály v jednotlivých RP (kromě signálů typu long line),
- žádné rekonfigurovatelné signály ve statické části návrhu,
- stejně vedené vodiče propojující statickou a rekonfigurovatelnou část obvodu.

Existuje celá řada (většinou akademických) postupů, jak vytvořit FPGA návrh s podporou techniky relokace hardwarových komponent. Podle způsobu úpravy a vytváření částečných konfiguračních souborů můžeme tyto techniky rozdělit na hardwarové a softwarové.

Příkladem hardwarových přístupů jsou např. aplikace REPLICA [35], REPLICA2PRo [36] a BiFR [12] – ve všech případech se jedná v podstatě o filtr, jehož vstupními daty je částečný konfigurační soubor uložený v paměti. Pokud je zapotřebí danou komponentu umístit na jiné místo v obvodu (relokovat), jsou tomuto filtru zadány parametry pro změnu pozice modulu a jako výstup je poskytnut nový částečný konfigurační soubor.

Mezi softwarové aplikace umožňující relokaci patří např. PARBIT [30] – jedná se o aplikaci vytvořenou v jazyce C, která jako vstupní data opět využívá informace uložené v původním konfiguračním souboru a jako výstup vrací soubor s daty odpovídajícími novému umístění dané komponenty. Na podobném principu je založena i aplikace PBITPOS [38].

Všechny zmíněné aplikace provádějí manipulaci s konfiguračním souborem ve smyslu změny obsahu FAR registru a změny kontrolního součtu CRC. Hlubší zásahy do konfiguračního souboru jsou prováděny např. v [31], [14] a [43]. V [31] se jedná o softwarovou aplikaci, která pracuje se vstupními daty v podobě původního konfiguračního souboru. Na rozdíl od aplikace PARBIT jsou vstupní data získávána vyčítáním z již nakonfigurovaného FPGA. V [43] se nejedná o manipulaci s konfiguračním souborem přímo za účelem jeho relokace, ale za účelem snížení potřebných implementačních kroků nutných k umístění dané komponenty na jinou pozici. To znamená, že z konfiguračního souboru jsou extrahovány informace o vnitřní logice a jejím propojení. Data, která mohou být beze změny použita na jiném místě v obvodu, se zachovávají, zbytek návrhu musí být znovu implementován. Popsané aplikace jsou vytvořeny pro starší typy FPGA (Virtex-E, Virtex II a Virtex II Pro a Virtex 4 v případě filtru BiFR). Ve všech případech je přemostění mezi statickou a dynamickou částí návrhu řešeno pomocí oddělovacího makra. Toto signálové rozhraní je firmou Xilinx podporováno pro obvody do řady Virtex 4 a návrhové prostření ISE do verze 9.2. Využití oddělovacího makra společně s podrobnou znalostí vnitřní struktury použitých obvodů a jejich konfiguračních souborů umožňuje výše zmíněným aplikacím snadné splnění podmínek nutných pro relokaci.

Ve [14] se autoři zabývají problematikou spojenou s heterogenním uspořádáním FPGA (problémové nalezení pozic vhodných pro relokaci) a možností relokace komponent mezi oblastmi neobsahujícími stejné logické zdroje. Tento problém je zde řešen tak, že logické zdroje, které nejsou obsaženy ve všech oblastech, nejsou v jednotlivých modulech vůbec využity (jejich použití je zakázáno již při implementaci návrhu) a při samotné relokaci nejsou tyto logické zdroje zahrnuty v modifikovaném konfiguračním souboru. K provádění hlubších zásahů do konfiguračního souboru u novějších obvodů (Virtex 5) je nezbytná dokonalá znalost jeho vnitřní struktury. Tato informace je ale firmou Xilinx považována za duševní vlastnictví a není pro návrháře dostupná. Technika relokace, popsaná v [14], byla vyvinuta ve spolupráci se společností Xilinx, autoři tedy měli detailní informace o skladbě konfiguračního souboru i celého obvodu.

Další možností, jak vytvořit návrh podporující relokaci, je s použitím volně dostupného nástroje zvaného GoAhead [15]. Tento nástroj je založen na kombinaci vlastního uživatelského rozhraní (GUI), textové verzi strukturního netlistu – XDL (Xilinx Design Language) a standardních implementačních nástrojů Xilinx. Při použití tohoto nástroje je nutná opakovaná konverze mezi textově založeným XDL formátem a binárním NCD (Native Circuit Description) formátem. Další nástroj, který podporuje tvorbu návrhu s možností relokace PB, je aplikace zvaná OpenPR [46]. Podobně jako předchozí aplikace i OpenPR využívá textové verze netlistů ve formátu XDL s tím rozdílem, že zde není k dispozici uživatelské rozhraní a aplikace je obsluhována pomocí skriptů z příkazové řádky. Obě tyto aplikace vykazují značnou složitost při návrhu a nejsou uživatelsky příliš přívětivé.

Jiný přístup k návrhu rekonfigurovatelného systému s podporou PBR je možné nalézt v [64], kde autoři popisují postup pro pevné umístění oddělovacích elementů a přesně definují propojení jednotlivých RP. Tento přístup umožňuje vytvoření oblastí s identickými vstupy a výstupy (stejný pinout). Popsaný postup je však manuální a pro větší komponenty nepoužitelný.

Bohužel i navzdory všem výhodám techniky PBR není vytvoření návrhů s podporou této techniky kompatibilní se signálovým rozhraním využívajícím oddělovacích elementů, a tudíž není v běžných návrhových nástrojích Xilinx dostupné.

Vzhledem k nutnosti využití techniky PBR bylo jedním z dílčích cílů práce vytvořit metodu umožňující vytvoření návrhu podporujícího relokaci. Pro zachování

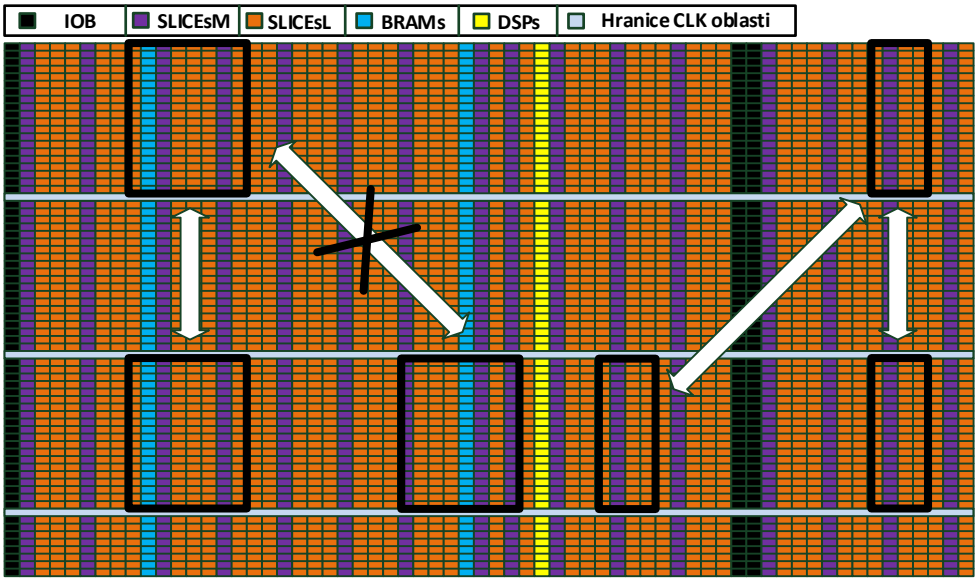
kompatibility této metody s co nejvíce obvody Xilinx (především z rodiny Virtex) bylo nutné, aby navržená metoda využívala běžných návrhových prostředků.

Vytvořená metodika je založená na úpravě standardního implementačního řetězce za účelem splnění jednotlivých podmínek nutných pro podporu techniky PBR. Relokace samotná je pak prováděna změnou adresy konfiguračních rámců částečného konfiguračního souboru.

4.1.1.1 Splnění podmínek nutných pro relokaci

Pro splnění prvních dvou podmínek (velikost oblasti, typ a množství vnitřní logiky) využijeme běžné omezující podmínky typu AREA_GROUP. Jedná se o implementační omezující podmínky, které se zapisují do souboru UCF. Tyto podmínky umožňují umístění jednotlivých částí návrhu do konkrétních fyzických oblastí v obvodu [59]. Příklad výběru vhodných oblastí je znázorněn na Obr. 24. Příklady použití omezujících podmínek AREA_GROUP jsou:

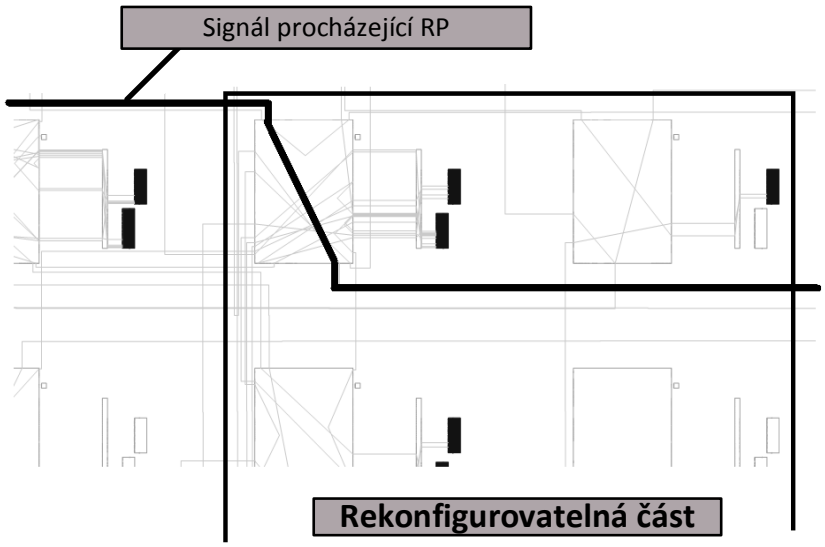
```
AREA_GROUP "název" RANGE=SLICE_XnYn:SLICE_XmYm;
AREA_GROUP "název" RANGE=DSP48_XnYn:DSP48_XmYm;
AREA_GROUP "název" RANGE=RAMB36_XnYn:RAMB36_XmYm;
```



Obr. 24: Příklad rozmístění logických zdrojů uvnitř FPGA a možnosti relokace

Zamezit vedení statických signálů skrz rekonfigurovatelné oblasti, tj. signálů, které využívá statická část systému a které rekonfigurovatelnou částí pouze procházejí (využívají propojovací matice, viz Obr. 25), lze několika způsoby. Je možné použít nástroj FPGA Editor a ručně tyto signály odstranit. Toto řešení je ale pro svou pracnost

a časovou náročnost vhodné pouze pro malý počet vodičů. Další možností je aplikace implementačních omezujících podmínek. V tomto případě je na výběr ze dvou typů omezení. Je možné využít omezující podmínky PRIVATE nebo ROUTING.



Obr. 25: Statický signál procházející skrz RP

Obě podmínky umožňují zakázat průchod statických signálů v rekonfigurovatelné části návrhu. Liší se v tom, že omezení ROUTING zakazuje průchod všech statických signálů skrz rekonfigurovatelnou oblast, zatímco omezení PRIVATE se týká pouze statických signálů, které využívají propojovacích matic uvnitř RP (např. vodičů typu long line křižujících RP se toto omezení netýká).

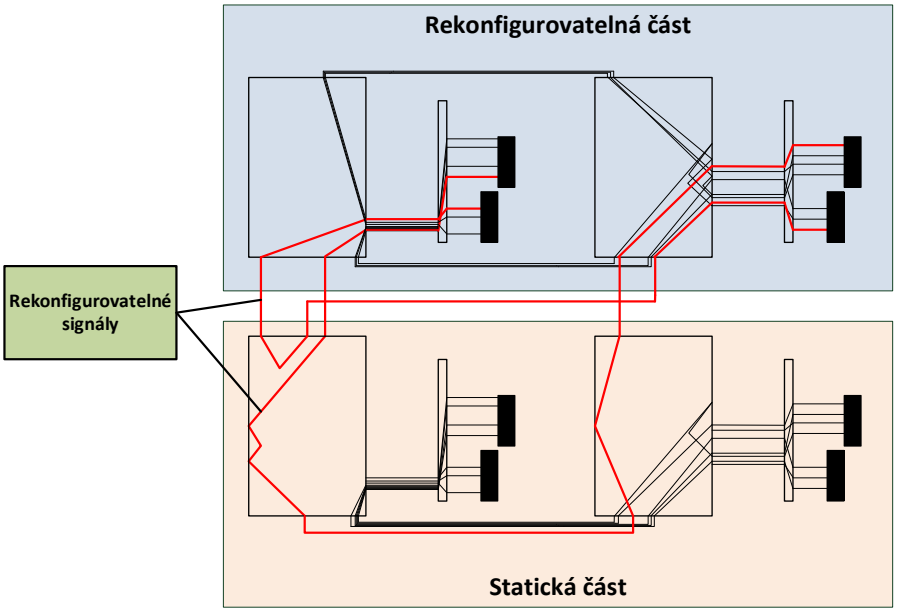
Vzhledem k tomu, že samotné použití částečné rekonfigurace způsobuje zhoršení časování statické části návrhu a snížení maximální rychlosti celého systému, je vhodnější využití omezující podmínky PRIVATE. Tyto omezující podmínky se zapisují do UCF souboru. Podrobnosti o těchto omezeních nejsou v dostupné (běžně dodávané) dokumentaci k dispozici. Příklad zápisu těchto do UCF souboru je:

```
AREA_GROUP "název" PRIVATE=ROUTE;
AREA_GROUP "název" ROUTING=CLOSED;
```

V dalším kroku je třeba zamezit signálům jednotlivých RP, aby zasahovaly do statické části návrhu. Popsaná situace je naznačena na Obr. 26. Tento problém jde řešit opět manuální cestou s pomocí softwaru FPGA Editor, tato metoda je však značně pracná. Nebo může být opět využita implementační omezující podmínka. Pro tento účel může posloužit omezení CONTAINED, které zajistí propojování rekonfigurovatelných

signálů pouze s využitím propojovacích zdrojů příslušné RP. Toto omezení se opět zapisuje do UCF souboru. Příklad zápisu je:

```
AREA_GROUP "název" CONTAINED=ROUTE;
```



Obr. 26: Rekonfigurovatelný signál využívající statických propojovacích prostředků

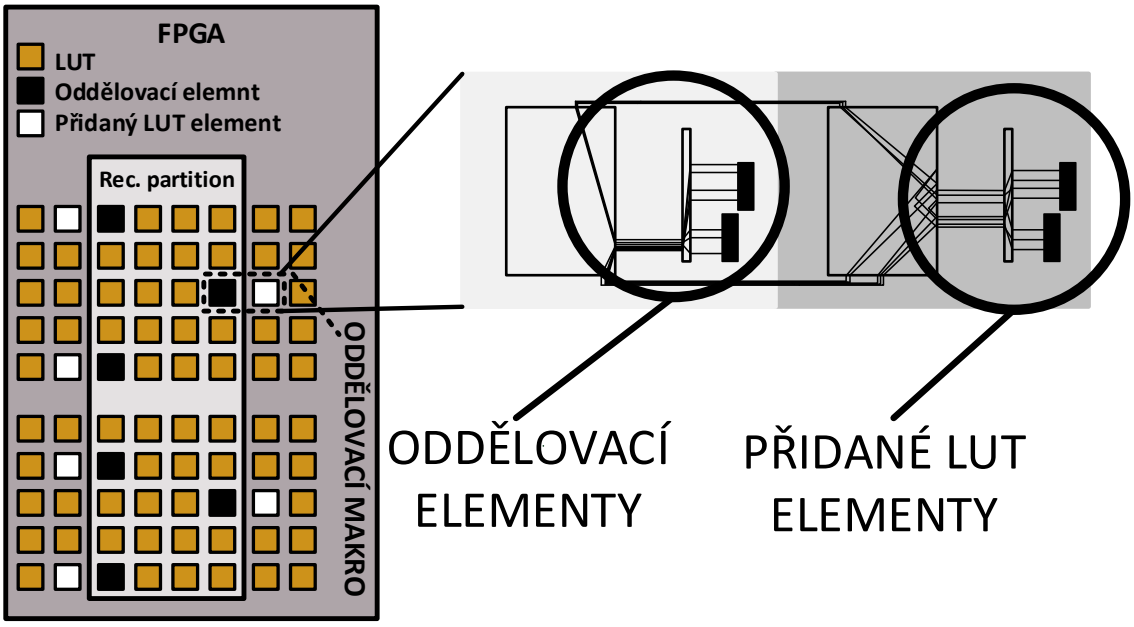
Poslední z obecných podmínek, které musí být splněny, aby bylo možné ve výsledném systému provádět relokaci, je striktní dodržení stejného propojení všech RP se statickou částí návrhu. Tato podmínka je hlavní příčinou toho, proč není technika PBR podporována stávajícími návrhovými nástroji, které využívají oddělovací elementy. Toto rozhraní je tvořeno pouze jedním LUT elementem na každý jeden bit signálu procházejícího ze statické části do té rekonfigurovatelné a naopak. Propojovací nástroj (PAR) připojuje individuálně každý oddělovací element, což způsobuje, že připojení oddělovacích elementů je v každém RP jiné (viz Obr. 19).

Jednou z možností, jak tento problém obejít, je opětovné použití nástroje FPGA Editor a manuální propojení jednotlivých signálů v celé jejich délce. Tento způsob je značně zdlouhavý a složitý (nepoužitelný pro běžný návrh). Jediným způsobem, kterým lze tedy docílit identického propojení jednotlivých signálů ve všech RP, je použití oddělovacího makra. Bohužel toto signálové rozhraní není ve stávajících návrhových nástrojích podporováno. Z tohoto důvodu je vytvořená metoda založena na myšlence přetransformování signálového rozhraní tvořeného pomocí oddělovacího elementu

(přístup podporovaný v aktuálních návrhových nástrojích) na přístup využívající rozhraní podobné dříve využívanému oddělovacímu makru.

Prvotní myšlenka jednoduše nahradit oddělovací element standardním oddělovacím makrem, tj. vložit oddělovací makro do návrhu jako tzv. hard makro pomocí souboru s příponou nmc stejně jako tomu bylo u starších návrhových nástrojů, se ukázala jako nereálná. Problém nastane po odstranění oddělovacích elementů, kdy implementační nástroje považují signály vstupující do RP za statické. Tento fakt koliduje s výše popsanými omezujícími podmínkami PRIVATE a CONTAINED a vedl ke zjištění, že oddělovací elementy musí být součástí vytvářených oddělovacích maker.

Vytvářené oddělovací makro je tedy tvořeno vždy z jednoho oddělovacího elementu a jednoho LUT elementu přidaného do návrhu před začátkem implementace. Tyto elementy jsou přidávány na úrovni HDL návrhu (do návrhu jsou přidávány instance komponenty LUT1). Ukázka návrhu v jazyce VHDL s přidanými LUT elementy je uvedena v Příloze A. Přidaný LUT element a oddělovací element jsou umístěny co nejblíže k sobě, aby bylo možné co nejjednodušeji kontrolovat jejich vzájemné propojení. Příklad umístění zmiňovaných LUT elementů tvořících vlastní oddělovací makro je vyobrazen na Obr. 27.



Obr. 27: Příklad umístění oddělovacích elementů a přidaných LUT elementů na hranici mezi statickou a rekonfigurovatelnou částí

Oddělovací elementy jsou do návrhu vkládány automaticky při překladu strukturního netlistu do nativní generické databáze implementačním nástrojem

NGDBUILD. Přidané elementy jsou do návrhu vloženy ještě před začátkem implementace. Pro vytvoření oddělovacího makra potřebujeme kontrolovat umístění těchto LUT elementů – to lze provádět pomocí LOC a BE*L* omezujících podmínek. K oddělovacím elementům přistupujeme při zápisu omezujících podmínek jako k pinům (oddělovací elementy nejsou před začátkem implementace v návrhu fyzicky přítomné), k přidaným LUT elementům jako k instancím. Příklad použití těchto omezení je:

```
PIN "nazev_oddelovaci_element" LOC=SLICE_XnYn;
PIN " nazev_oddelovaci_element " BEL=A6LUT;
INST "nazev_vlozeny_LUT_element" LOC=SLICE_XnYn;
INST "nazev_vlozeny_LUT_element" BEL=A6LUT;
```

Popsaný postup vkládání a umisťování jednotlivých LUT elementů bylo kvůli značné náročnosti nezbytné zautomatizovat. Z tohoto důvodu byl vytvořen skript, který vkládá "přidávané" LUT elementy a zajišťuje jejich propojení se zbytkem návrhu. Dále byl vytvořen skript, který vytváří omezující UCF podmínky potřebné pro umístění jednotlivých elementů.

Vkládání LUT elementů do návrhu provádí skript *my_macro_insertion*. Skript ke své činnosti vyžaduje VHDL popis části návrhu hierarchicky nadřazený komponentám, které mají být relokovány. Skript načte ze zdrojového souboru všechna potřebná data a vytvoří nový VHDL soubor, který je doplněný o přidávané LUT elementy a jejich propojení. Skript provádí následující operace:

- ve vstupním VHDL souboru je vyhledána komponenta určená k relokaci,
- piny nalezené komponenty včetně propojení jsou uloženy do paměti,
- podle počtu nalezených pinů je vytvořena komponenta *my_macro.vhd*, do které jsou vloženy přidávané LUT elementy pro všechny piny komponenty kromě pinů určených pro hodinové signály (CLK signály neprochází skrz žádné signálové rozhraní),
- dojde k propojení komponenty *my_macro* se zbytkem návrhu,
- provedené úpravy jsou spolu s obsahem původního VHDL souboru vloženy do nového VHDL souboru použitelného k následné implementaci.

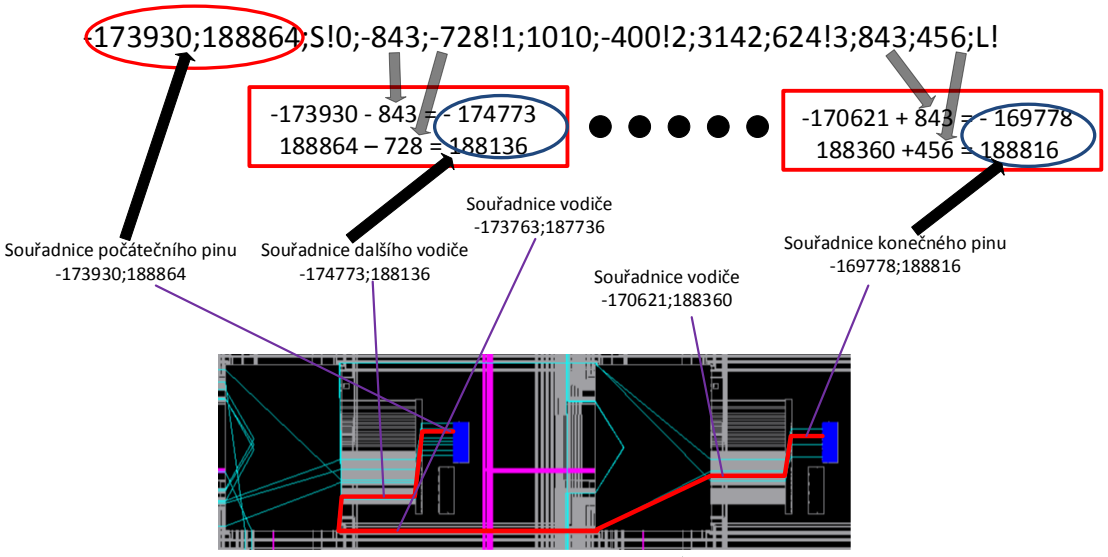
Omezující podmínky, potřebné pro umístění LUT elementů (tvořících oddělovací makro) generuje skript *macro_position*. Tento skript ke své činnosti vyžaduje stejný VHDL popis jako v předchozím případě a informaci o požadovaném umístění dané komponenty v obvodu. Výstupem skriptu je UCF soubor, kde jsou uloženy omezující podmínky s požadovanými souřadnicemi. Skript provádí následující operace:

- ve vstupním VHDL souboru je vyhledána komponenta určená k relokaci,
- z názvů jednotlivých pinů komponenty jsou odvozeny pozice pro umístění oddělovacích elementů, které jsou následně spárovány s LUT elementy z komponenty *my_macro*,
- pro přehlednost jsou vytvořená oddělovací makra rozdělena na vstupní a výstupní a podle tohoto kritéria jsou umístěna na levou nebo pravou hranici komponenty (tj. hranici mezi statickou a dynamickou částí návrhu),
- je vytvořen výstupní soubor *system.ucf*.

Propojení mezi jednotlivými LUT elementy (oddělovací element a přidaný LUT element) je kontrolováno pomocí tzv. DIRT (Direct Routing Constraints) omezujících podmínek. Vzhledem k umístění obou LUT elementů blízko u sebe kontrolujeme pouze propojení krátkých úseků (viz Obr. 29). DIRT omezení jsou v dostupné dokumentaci popsána velice stroze. Příklad jejich použití je:

```
NET"nazev_signalu"ROUTE="{-173930;188864;S!0;-843;-728!1;1010;-
400!2;142;624;L!}";
```

Signálové vodiče jsou při použití DIRT adresovány pomocí souřadnicového systému. První dvě čísla (souřadnice) představují absolutní adresu zdroje signálu (konkrétní pin, kde daný signál začíná). Ostatní čísla udávají relativní adresy dalších vodičů, kudy signál prochází, přičemž jednotlivé souřadnice jsou přičítány a odčítány od původní absolutní adresy začátku signálu. Toto adresování je naznačeno na Obr. 28.



Obr. 28: Adresy vodičů v obvodu při použití DIRT

DIRT omezení se zapisují přímo do UCF souboru. Pokud máme detailní přehled o daném obvodu (tj. známe souřadnice jednotlivých vodičů), je možné použít DIRT již během první implementace. Pokud detailní informace neznáme, je možné návrh nejprve částečně implementovat a z vytvořeného netlistu získat informace o propojení jednotlivých RP.

Z takto získaných informací je možné vytvořit DIRT omezení, která zajistí identické propojení statické části s rekonfigurovatelnými oblastmi. Vzhledem k úmyslu docílit co největší univerzálnosti popisované metody relokace se tento postup při práci s rekonfigurovatelnými systémy ukázal jako jednodušší a rychlejší.

Stejně jako v předchozím případě není možné vytvářet DIRT omezení manuálně (obzvlášť při větším množství pinů). Bylo tedy nutné vytvořit skript, který požadované omezující podmínky generuje. Tento skript je pojmenován *routing*. Vstupem skriptu je UCF soubor s DIRT podmínkami vzniklými při první implementaci (informace získané s využitím nástroje FPGA Editor). Výstupem je opět UCF soubor, který obsahuje DIRT omezení, která zajišťují stejné propojení u všech oblastí. Skript provádí následující operace:

- ze vstupního UCF souboru jsou vyčteny názvy všech potřebných signálů spolu s absolutní adresou jejich začátku,
- k absolutním adresám jednotlivých signálů jsou přidány zbylé relativní souřadnice tak, aby propojení všech oblastí bylo shodné,
- je vytvořen výstupní UCF soubor.

I navzdory použití DIRT podmínek může stále docházet k problémům s dodržením podmínky o shodném propojení mezi SP a všemi RP, a to zejména u RM s velkým počtem vstupních a výstupních pinů. To je způsobeno tím, že propojování různých signálů je prováděno s různou prioritou (některé signály jsou propojeny dříve než ostatní). Tento fakt může způsobit, že propojení signálu zadaného pomocí DIRT podmínky není možné, protože daná cesta (vodič, část propojovací matice) je již zabraná. V těchto případech propojovací nástroj (PAR) tuto DIRT podmínku ignoruje a propojí signál jinudy.

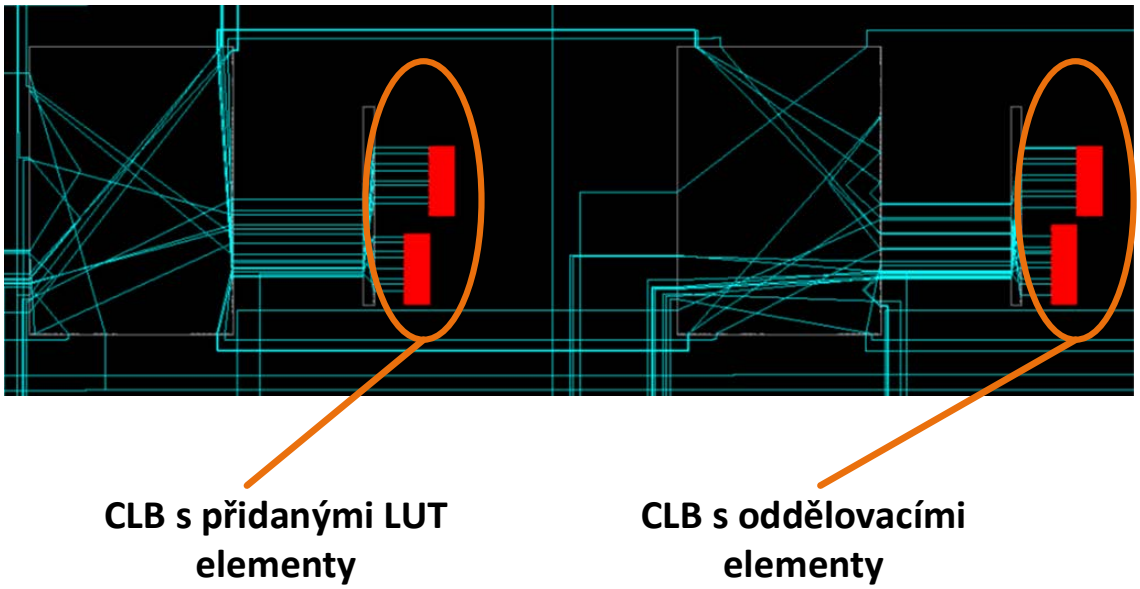
Tomuto problému lze předejít použitím podmínky, jejímž prostřednictvím lze stanovit prioritu propojování jednotlivých signálů. Jedná se o omezující podmínku PRIORITIZE, která se zapisuje do PCF souboru a která umožňuje nastavení priority propojování v rozmezí 1 (nejnižší priorita) až 100 (nejvyšší priorita). Příklad této omezující podmínky, která opět není popsaná v dodávané dokumentaci, je (v tomto tvaru je signálu přiřazena nejvyšší priorita):

```
NET nazev_signalu PRIORITIZE = 100;
```

Nastavení priority propojování je opět prováděno pomocí skriptu. Jedná se o nadstavbu skriptu routing, který po aktivaci volby priorita z načteného jmenného seznamu všech signálů vytvoří požadovaný PCF soubor.

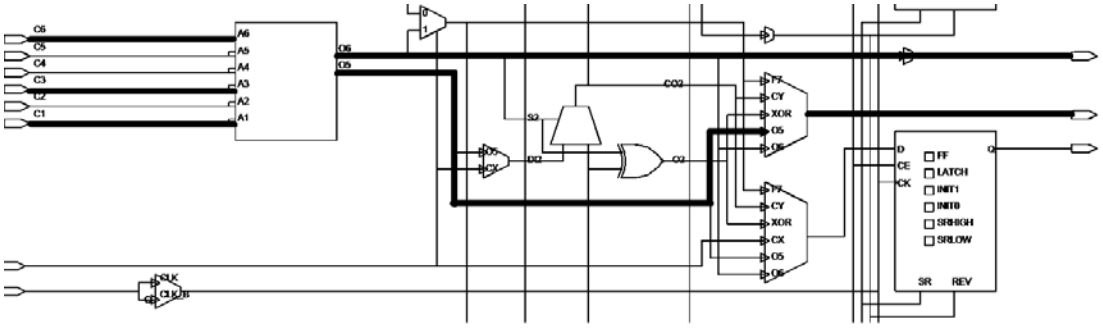
4.1.1.2 Přístup pro snížení dodatečného hardwarového vybavení

Popsaná metoda pro relokaci způsobuje dodatečné navýšení potřebného hardwarového vybavení (logický overhead) o jeden LUT element na každý bit signálu procházejícího přes hranici mezi statickou a rekonfigurovatelnou částí v porovnání s rekonfigurací založené na oddělovacích elementech. To je graficky znázorněno na Obr. 29, kde pro každé jedno CLB využité pro oddělovací elementy potřebujeme jedno CLB s přidanými LUT elementy.



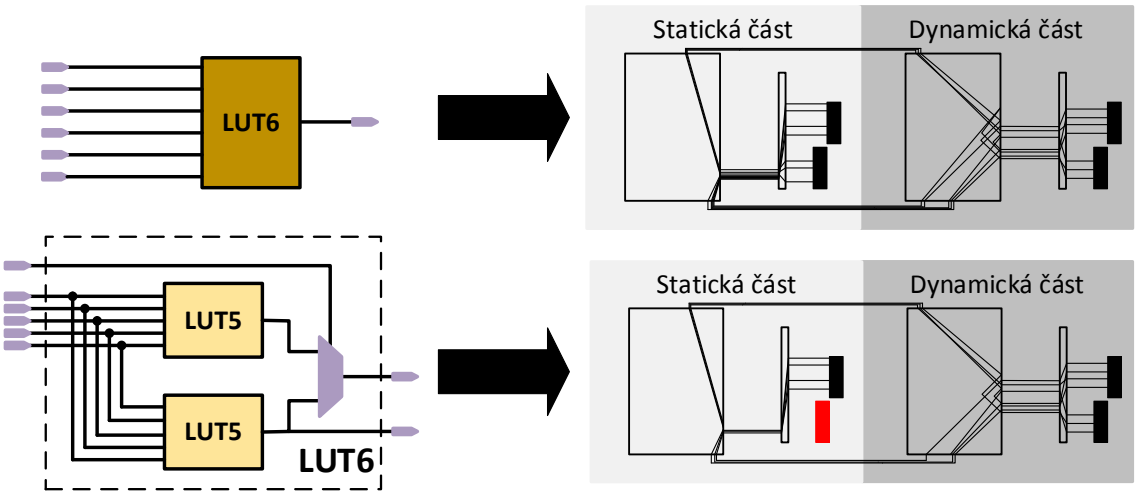
Obr. 29: Propojení oddělovacího makra (využity všechny 4 CLB)

Za účelem snížení dodatečného navýšení potřebné logiky lze využít obou výstupů u přidaných LUT elementů (Obr. 30), což ve skutečnosti znamená využití dvou pětivstupových LUT elementů na místo jednoho šestivstupového (Obr. 31). Tím lze dosáhnout režie jeden LUT element na každé dva bity signálu procházejícího přes hranici mezi oběma oblastmi. Na Obr. 32 můžeme vidět, že s využitím tohoto přístupu potřebujeme dva CLB přidaných LUT elementů na čtyři CLB s oddělovacími elementy.



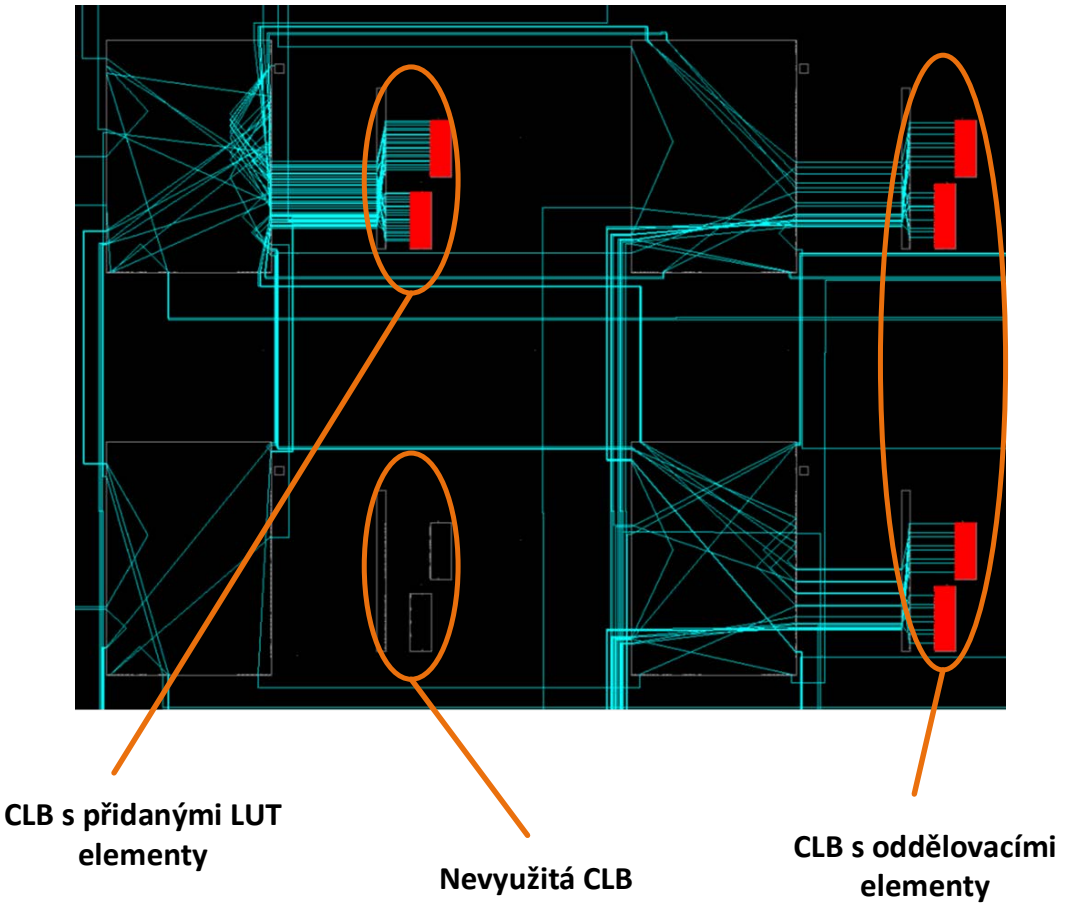
Obr. 30: Ilustrace využití obou výstupů LUT elementu

Využití stejného principu u oddělovacích elementů za účelem dalšího snížení dodatečné logiky není bohužel možné, jelikož tyto LUT elementy nejsou před začátkem implementace v návrhu fyzicky přítomny, a tudíž na ně není možné aplikovat stejná nastavení jako na přidané LUT elementy.



Obr. 31: Použití dvou 5vstupových LUT elementů

Popsaný přístup pro snížení množství dodatečného hardwarového vybavení zvyšuje náročnost propojování jednotlivých oblastí a v některých případech může zhoršit časování daného rekonfigurovatelného modulu. Proto je vhodné tento přístup využívat pouze u aplikací citlivých na množství spotřebované logiky.



Obr. 32: Propojení oddělovacího makra se sníženou režií (využity pouze 3 CLB)

4.1.1.3 Další úskalí

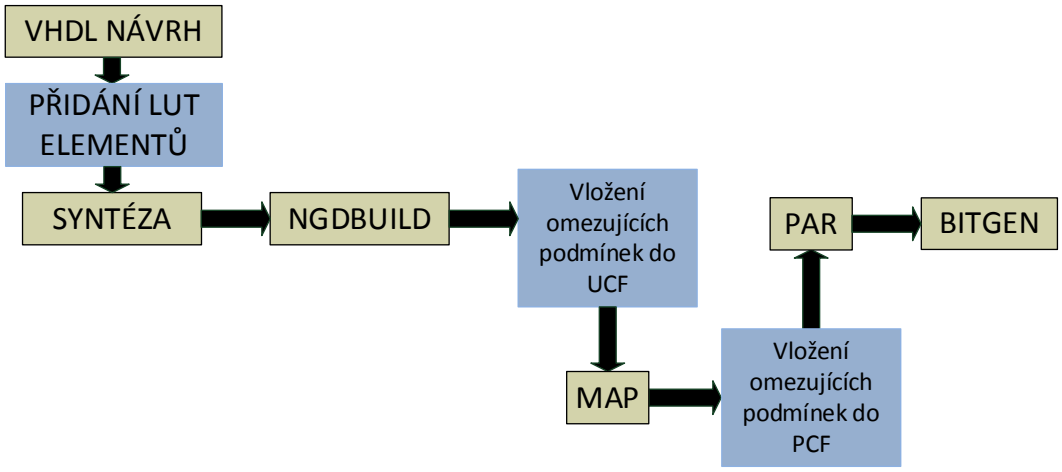
Při úpravě konfiguračního souboru za účelem jeho relokace (tj. změna adresy konfiguračního rámce) dochází ke změně ve struktuře tohoto souboru a je zapotřebí přepočítat kontrolní součet CRC nebo jeho kontrolu úplně zakázat. Kontrolu CRC lze nastavit přímo v nástroji BitGen, a to zadáním parametru: *-g crc:disable*.

Vzhledem k tomu, že výrobce používaného FPGA uvádí, že pravděpodobnost zničení obvodu nahráním vadného konfiguračního souboru je velmi malá, byly pro jednoduchost všechny experimenty v rámci této práce prováděny bez používání kontrolního CRC součtu.

4.1.2 Přehled zásahů do implementačního řetězce

Popsaná relokační metodika je založena na transformaci přístupu využívajícího oddělovacích elementů na přístup využívající oddělovací makra. Tato transformace je realizována pomocí implementačních omezujících podmínek. Pro realizaci systému podporujícího relokaci bylo nezbytné upravit běžný implementační řetězec (kapitola 1.3). Změny standardního implementačního řetězce jsou ilustrovány na Obr. 33.

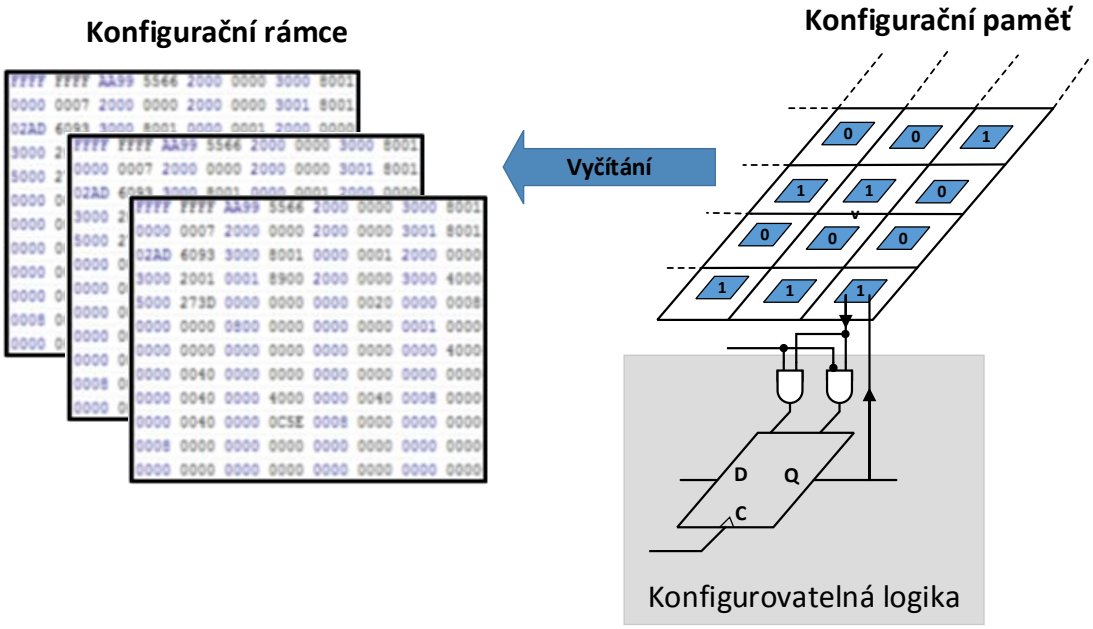
Proces implementace je doplněn o krok, kdy jsou na úrovni VHDL návrhu přidány LUT elementy tvořící polovinu oddělovacího makra. Takovýto doplněný návrh poté projde standardní syntézou. Během překladu nástrojem NGDBILD jsou do návrhu automaticky vloženy oddělovací elementy. Před spuštěním mapování (MAP) jsou do návrhu vloženy omezující podmínky zapisované do UCF – jedná se o omezující podmínky pro kontrolu velikosti a polohy rekonfigurovatelných oblastí (AREA_GROUP, RANGE) a omezující podmínky pro kontrolu polohy jednotlivých LUT elementů (LOC, BEL). Dále jsou do UCF souboru vkládány omezující podmínky pro odstranění statických signálů z rekonfigurovatelné části systému (PRIVATE) a naopak (CONTAINED). Jako poslední jsou zapisovány DIRT omezující podmínky. Ač se jedná o podmínky kontrolující propojení, jsou uváděny v UCF souboru, tedy před spuštěním mapování. Po namapování do daného FPGA obvodu jsou do návrhu přidány omezující podmínky pro kontrolu priority (podmínka PRIORITIZE) propojení vodičů spojujících jednotlivé LUT elementy tvořící oddělovací makro.



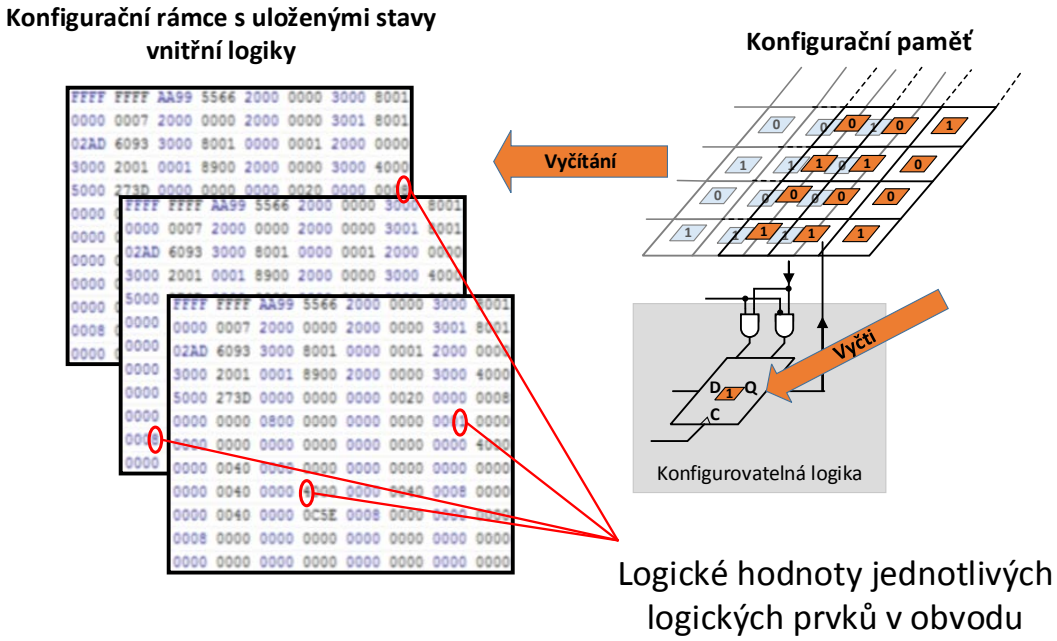
Obr. 33: Upravený implementační řetězec

4.2 Zpětné vyčítání konfigurační paměti

Další technikou, jejíž využití je pro tuto práci nezbytné, je zpětné vyčítání konfigurační paměti. Tato technika umožňuje vyčtení všech dat z interní konfigurační paměti FPGA obvodu. Zpětné vyčítání konfigurační paměti rozdělujeme na dva základní typy. Prvním z nich je tzv. Readback Verify, s jehož využitím lze provádět ověření správnosti dat uložených v konfigurační paměti obvodu. Druhým typem je tzv. Readback Capture – tento typ se používá pro vyčítání aktuálního stavu všech paměťových elementů (LUT elementy, klopné obvody, IOB registry, distribuovaná RAM paměť a BRAM paměť) v obvodu. Takto vyčtená data se využívají při odlaďování návrhu a funkční verifikaci [52], [65].



Obr. 34: Blokové naznačení techniky Readback Verify



Obr. 35: Blokové naznačení techniky Readback Capture

Na Obr. 34 a Obr. 35 je blokově naznačen princip obou zmíněných typů techniky zpětného vyčítání konfigurační paměti. Vyčítání za účelem kontroly správnosti konfiguračních dat (Obr. 34) umožňuje v libovolném okamžiku získat všechna konfigurační data (tj. data určující funkci veškeré interní logiky a její propojení). Tato technika je v podstatě základem pro jednoduchý test konfigurační paměti, který se nazývá bitstream (readback) scrubbing [18], [20], [29], [45]. Při použití readback scrubbing probíhá periodické vyčítání konfigurační paměti obvodu. Tato data se porovnávají s konfiguračním souborem uloženým v paměti chráněné (např. stíněním) proti účinkům radiace (tzv. golden bitstream). Při zjištění výskytu poruchy (tj. rozdílu mezi vyčtenou a uloženou hodnotou) dojde k přepsání konfigurační paměti obvodu. Někdy je vypočítán pouze kontrolní součet namísto všech konfiguračních dat, výsledek tohoto součtu je pak porovnáván s referenční hodnotou.

Druhý typ zpětného vyčítání (Obr. 35) není principiálně nijak odlišný od toho prvního. Zásadním rozdílem je, že po zadání příkazu pro zaznamenání stavů (capture) jsou stavy všech vnitřních registrů uloženy do nevyužitých částí konfigurační paměti. Po přečtení obsahu této paměti získáme spolu s konfiguračními daty i hodnoty vnitřních stavů všech vyčítaných registrů.

Při provádění prvního typu zpětného vyčítání nejsou na zkoumaný systém kladeny žádné zvláštní požadavky a tuto techniku je možné provádět prakticky bez jakéhokoli omezení a funkčnost systému není nijak ovlivněna. Nejinak je tomu i u typu druhého,

avšak chceme-li získat plnohodnotná data, nesmí během procesu vyčítání docházet ke změnám v paměťových prvcích v obvodu (např. zápis do BRAM atd.).

4.2.1 Implementace zpětného vyčítání konfigurační paměti

Podpora techniky vyčítání konfigurační paměti v rekonfigurovatelném systému nejenže přináší zvýšení flexibility, ale umožňuje další snížení paměťových nároků (některé částečné konfigurační soubory lze vyčíst z konfigurační paměti) a zvyšuje provozuschopnost a spolehlivost celého systému (např. technika readback scrubbing, tvorba kontrolních bodů atd.).

Pro zavedení techniky zpětného vyčítání konfigurační paměti do systému je třeba nastavit generátor konfiguračních souborů (BitGen) tak, aby vytvořené soubory umožňovaly přístup k jejich datům (nezbytné pro DPR a zpětné vyčítání). Tuto funkci nastavuje parametr security. Dále je třeba povolit zpětné vyčítání konfigurační paměti parametrem readback. Tyto dva parametry jsou dostačující při zpětném vyčítání přes rozhraní ICAP. Zmíněné parametry generátoru konfiguračních souborů lze zadat v sekci nastavení konfigurace (Set Configuration) a při zápisu např. do příkazové řádky jsou uvozeny pomocí parametru *-g*. Způsob zápisu je následující:

-g Security: none
-g ReadBack: yes

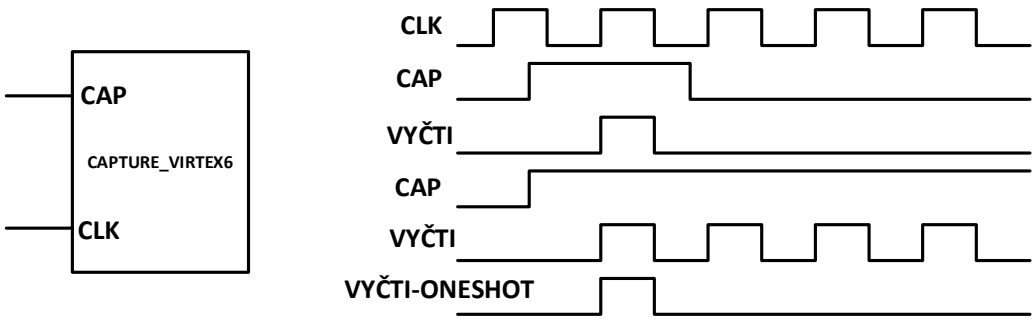
Při provádění techniky zpětného vyčítání pomocí rozhraní SelectMAP je třeba nastavit ještě následující dva parametry. Jedná se o parametr persist, který zakazuje přístup k nastavovacímu (mode) pinu tohoto rozhraní. Dále musí být zakázáno šifrování konfiguračního souboru nastavením parametru encrypt. Absence schopnosti šifrování velice snižuje možnosti využití této techniky pomocí portu SelectMAP v praktických aplikacích z důvodu nemožnosti zabezpečení konfiguračních souborů. Zpětné vyčítání konfigurační paměti je možné provést i skrze rozhraní JTAG, které za tímto účelem využívají některé verifikační nástroje. Způsob zápisu dalších parametrů je následující:

-g Persist: yes
-g Encrypt: no

Pokud jsou generátoru konfiguračních souborů nastaveny výše popsané parametry, po dokončení implementace je k dispozici vše potřebné k provádění prvního typu zpětného vyčítání (typ verify). Mimo požadovaných konfiguračních souborů jsou k dispozici tzv. maskovací soubory určené k maskování bitů ve vyčteném datovém rámci. Maskují se např. bity reprezentující obsah uživatelské paměti, protože se nejedná o konfigurační data. Tyto bity tedy nemohou být při kontrole do porovnání zahrnuty. Samotnou verifikaci vyčtených konfiguračních dat lze provádět např. s využitím softwarového nástroje Xilinx IMPACT, který je standardní součástí balíku návrhových nástrojů Xilinx ISE. Podrobnější informace o verifikaci konfiguračních dat s využitím zpětného vyčítání konfigurační paměti a o maskovacích souborech k této technice potřebných je k nalezení např. v [65].

Druhý typ zpětného vyčítání (typ capture) je jakousi nadstavbou typu prvního, kdy jsou vyčtená konfigurační data doplněna o stavy interních registrů, které se ukládají do paměti v přesně definovaných okamžicích, po zadání příkazu vyčti (capture). Nutnost zadání příkazu pro vyčtení je vlastně jedním z hlavních rozdílů mezi těmito technikami. Zadání příkazu pro záznam v podstatě znamená zapsání příslušného příkazu do konfiguračního registru FPGA obvodu. K uložení vnitřních stavů dochází vždy s následující náběžnou hranou systémových hodin po zapsání příkazu do registru.

Zadání příkazu pro uložení obsahu interních registrů do paměti je možné provést dvěma způsoby. První možností, se kterou je třeba uvažovat již při návrhu, je vložit do systému komponentu zvanou CAPTURE_VIRTEX6. Pokud na vstup označený jako CAP (capture) přivedeme log 1, tak na následující takt hodinového signálu CLK dojde k uložení všech vnitřních stavů. K uložení vnitřních stavů bude docházet na každou náběžnou hranu signálu CLK, pokud zůstane signál CAP v log. 1. Tomuto vícenásobnému ukládání vnitřních stavů obvodu lze zabránit nastavením log 0 na vstupu CAP po prvním uložení hodnot nebo lze komponentě CAPTURE_VIRTEX6 nastavit atribut *ONESHOT*. Tato volba zajistí, že vnitřní stavy logiky se do konfigurační paměti uloží pouze na první náběžnou hranu signálu CLK. Blokové schéma komponenty CAPTURE_VIRTEX6 společně s naznačenými průběhy generování příkazu pro výčet je vyobrazeno na Obr. 36.



Obr. 36: Blokové schéma komponenty pro provedení příkazu *vyčti* s naznačeným průběhem tohoto příkazu.

Druhou možností, jak provést příkaz capture, je jeho zapsání přímo do konfiguračního registru obvodu skrze konfigurační port. Pro vykonání příkazu je třeba do konfigurační logiky zapsat sekvenci konfiguračních slov, po které se příkaz provede. Tato sekvence je popsána v Tab. 2.

Tab. 2: Sekvence konfiguračních slov pro vykonání příkazu CAPTURE

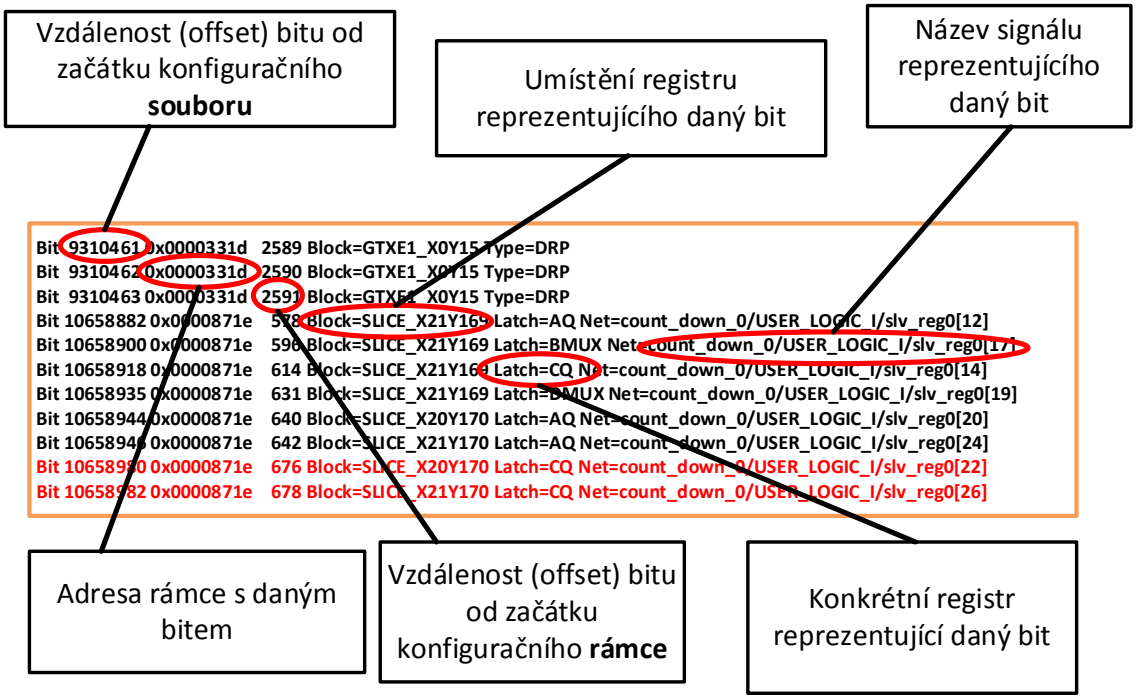
Konfigurační data – hex formát	Vysvětlení příkazu
FFFFFFFF	vyplňující slovo
AA995566	synchronizační slovo
20000000	prázdný příkaz
30008001	požadavek zápisu jednoho konfiguračního slova do CMD registru
0000000C	příkaz CAPTURE
FFFFFFFF	vyplňující slovo
FFFFFFFF	vyplňující slovo

K tomu, abychom mohli určit, kterému internímu registru odpovídá který bit ve vyčtených datech, se využívá tzv. alokační soubor konfigurační paměti (memory allocation file). V tomto souboru je uložená informace o tom, kde v obvodu (konkrétní SLICE) se daný registr nachází a který bit v konfiguračním rámci odpovídá vnitřní hodnotě tohoto registru.

Alokační soubor je vytvářen generátorem konfiguračních souborů, pokud je s parametrem *-g readback* nastaven ještě parametr *-b*, kterým je možné vytvořit i tzv. rawbitstream (soubor s příponou rbt). Jedná se o konfigurační soubor, který na rozdíl od standardního binárního konfiguračního souboru (soubory s příponou bit) obsahuje konfigurační data ve formátu ASCII.

Alokační soubor konfigurační paměti je textový soubor s informacemi o tom, který bit v konfiguračním rámci odpovídá určitému bitu v konfigurační paměti FPGA obvodu. Ukázka tohoto souboru je vyobrazena na Obr. 37, kde každý řádek představuje

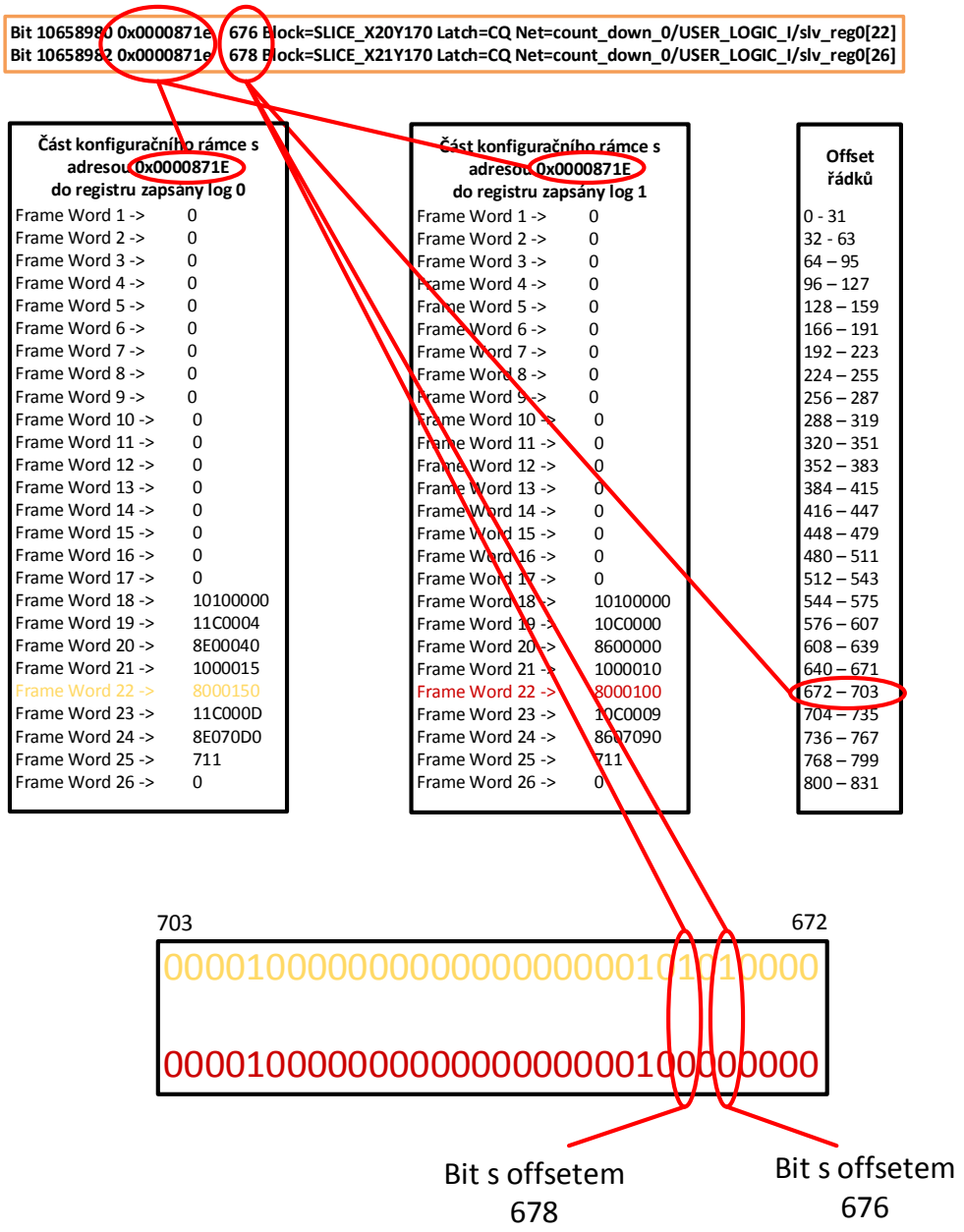
informace o jednom bitu v konfigurační paměti. První číslo v každém řádku představuje offset (vzdálenost v bitech) od začátku konfiguračního souboru, druhé číslo je adresa konfiguračního rámce, ve kterém se daný bit nachází. Dalším údajem je offset tohoto bitu od začátku konfiguračního rámce (určeného předcházející adresou). Další údaj vypovídá o tom, kde v obvodu a v jakém typu vnitřní logiky se daný registr (jenž je reprezentován vyčítaným bitem) nachází. Poslední dva údaje říkají, o jaký typ klopného obvodu se jedná a jak je v systému pojmenován.



Obr. 37: Výřez z alokačního souboru konfigurační paměti s popisem jednotlivých částí

To, jak jednotlivé údaje v alokačním souboru souvisí s vyčteným konfiguračním rámcem, je dále ilustrováno na příkladu záznamu obsahu registru pojmenovaného slv_reg0. Konkrétně se jedná o 23. a 26. bit v tomto registru – červeně zvýrazněné řádky na Obr. 37.

Na Obr. 38 je vyobrazena část konfiguračního rámce získaná pomocí zpětného vyčítání. První sloupec zobrazuje data konfiguračního rámce pro případ, kdy byly do registru slv_reg0 na všechny pozice zapsány log 0. Druhý sloupec zobrazuje týž rámec, s tím rozdílem, že do registru slv_reg0 byly na všechny pozice zapsány log 1. Ve třetím sloupci jsou rozepsány vzdálenosti jednotlivých řádků (offset) od začátku daného konfiguračního rámce. Obr. 38 dále naznačuje souvislost mezi obsahem alokačního souboru a bity v konfiguračním rámci.



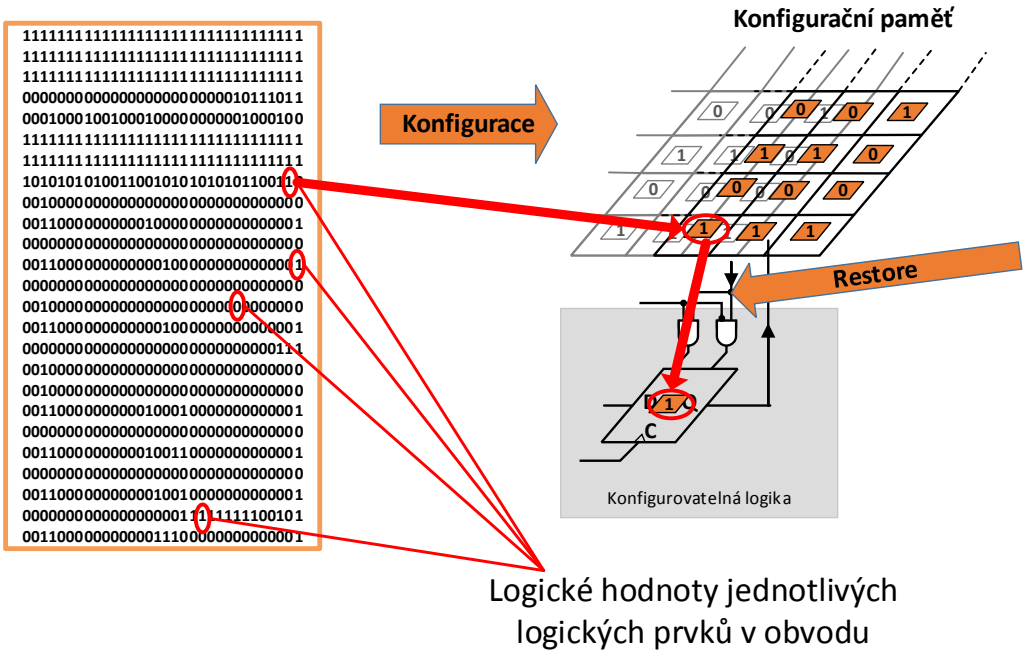
Obr. 38: Souvislost alokačního souboru konfigurační paměti s bity v konfiguračním rámci

Z výše popsaného příkladu vyplývá, že vyčtené stavy všech registrů jsou zobrazeny ve své inverzní podobě. S tímto faktem je třeba počítat i v případech, kdy provádíme změny přímo v konfiguračním souboru.

Použití techniky zpětného vyčítání konfigurační paměti není v dodávané dokumentaci příliš dobře popsáno. Sám výrobce uvádí, že není třeba poskytovat detailní informace, protože pro funkční verifikaci a odlaďování navrhovaných systémů jsou k dispozici dodávané softwarové nástroje.

4.3 Zápis dat do interních registrů FPGA obvodu

Třetí dílčí technikou, na jejímž využití je tato práce založena, je možnost zápisu dat do interních registrů FPGA obvodu neboli tzv. restore. Hodnota každého logického prvku schopného uchovat logickou informaci (registr) v obvodu FPGA je v konfigurační paměti obvodu reprezentována jedním bitem (INT0/INT1). Stejně jako technika zpětného vyčítání umožňuje získání hodnoty těchto bitů, tak technika zápisu dat umožňuje opačný přístup, kdy je hodnota těchto bitů reprezentujících vnitřní stavy zapsána do FPGA obvodu. Tento proces je principiálně naznačen na Obr. 39.



Obr. 39: Princip techniky zápisu dat do interních registrů

Technika zápisu dat do interních registrů umožňuje rozšíření rekonfigurovatelného systému o následující vlastnosti:

- vkládání dat na libovolné místo v systému,
- relokace komponent včetně jejich vnitřních stavů,
- synchronizace po rekonfiguraci (možné využití např. u systémů TMR),
- přesun funkce jednoho modulu do jiného modulu (pokud jsou tyto moduly stejné).

4.3.1 Implementace zápisu do interních registrů obvodu

Přepsání hodnot interních registrů se provádí zadáním příkazu: zapiš (restore), kdy dojde k nastavení signálu GSR (Global Set/Reset) a tím k aktualizaci všech bitů (reprezentujících vnitřní hodnoty) v konfigurační paměti.

Zadání příkazu pro zápis je možné provést více způsoby. První možností je již při návrhu vložit do systému komponentu STARTUP, která umožňuje nastavení signálu pro vykonání funkce zápisu do interních registrů (signál GSR).

Druhou možností, jak zápis do registrů provést, je zapsáním příkazu GRESTORE přímo do řídicího (CMD) registru FPGA obvodu. Vždy po zapsání tohoto příkazu dojde k přepsání vnitřních stavů všech klopných obvodů. Pro vykonání příkazu GRESTORE je třeba zadat sekvenci konfiguračních slov uvedených v Tab. 3.

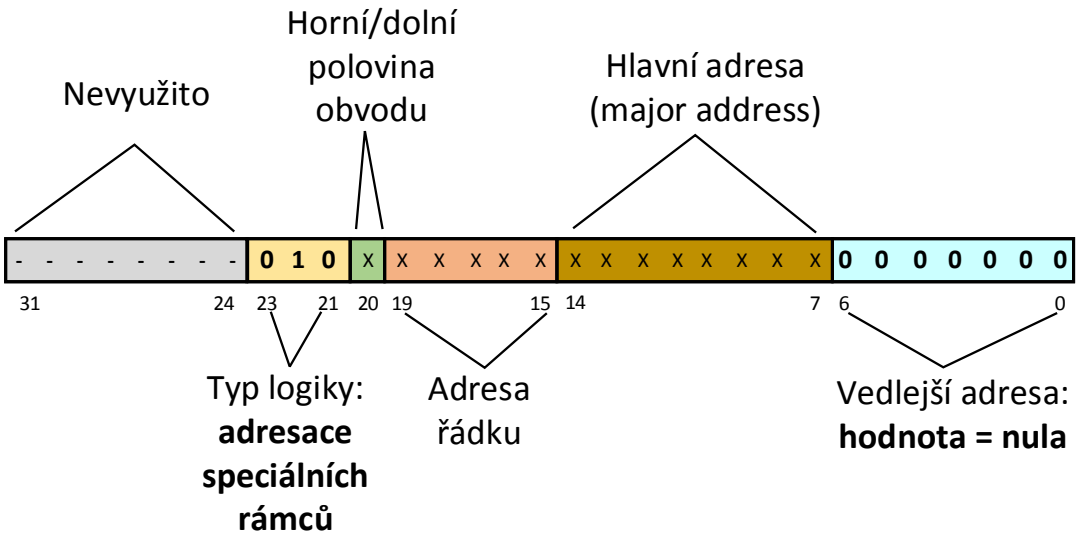
Pokud by techniku zapisování dat do interních registrů nebylo možné omezit na přepsání pouze některých částí obvodu, stala by se prakticky nepoužitelnou pro využití v částečně rekonfigurovatelném systému (při přepsání celého obvodu mluvíme o rekonfiguraci celkové, nikoli částečné).

Omezení této techniky pouze na některé části obvodu lze provést s využitím tzv. maskovacího bitu jednotlivých rekonfigurovatelných rámců. Každý sloupec logických prvků v obvodu má k dispozici tento maskovací bit. Rekonfigurovatelný rámec, který má tento bit nastaven, nemůže být ovlivněn technikami zpětného vyčítání konfigurační paměti ani zápisem dat do interních registrů. Maska částí obvodu, které nemají být ovlivněny, musí být nastavena před použitím těchto technik.

K maskovacímu bitu lze přistupovat pomocí speciálních (skrytých) konfiguračních rámců. Každému sloupci logiky v CLK regionu přísluší jeden tento speciální konfigurační rámec. Adresa tohoto rámce má stejný formát jako adresy ostatních konfiguračních rámců (kapitola 1.2). To znamená, že hlavní (major) adresa odpovídá adrese sloupce, který má být maskován, vedlejší (minor) adresa musí být nulová a sekce adresy rámce určující typ adresované logiky musí být nastavena na adresaci speciálních rámců, tj. sekce typ logiky (block type) musí odpovídat 010. Formát adresy pro přístup k těmto rámcům je naznačen na Obr. 40. Maskování jednotlivých rekonfigurovatelných rámců se provádí pomocí 41. konfiguračního slova. Toto slovo je určené pro nastavení hodinových HROW řádků umístěných v každém CLK regionu (viz Obr. 10).

Tab. 3: Sekvence konfiguračních slov pro vykonání příkazu RESTORE

Konfigurační data – hex formát	Popis příkazu
FFFFFFFF	vyplňující slovo
FFFFFFFF	vyplňující slovo
000000BB	požadavek pro zápis do registru pro nastavení šířky sběrnice
11220044	nastavení šířky sběrnice
FFFFFFFF	vyplňující slovo
FFFFFFFF	vyplňující slovo
AA995566	synchronizační slovo
20000000	prázdný příkaz
30008001	požadavek zápisu jednoho konfiguračního slova do CMD registru
00000007	příkaz pro reset CRC registru
20000000	prázdný příkaz
20000000	prázdný příkaz
30008001	požadavek zápisu jednoho konfiguračního slova do CMD registru
04250093	identifikační kód obvodu
30008001	požadavek zápisu jednoho konfiguračního slova do CMD registru
0000000B	vypínací příkaz
20000000	prázdný příkaz
30008001	požadavek zápisu jednoho konfiguračního slova do CMD registru
00000007	příkaz pro reset CRC registru
20000000	prázdný příkaz
20000000	prázdný příkaz
20000000	prázdný příkaz
20000000	prázdný příkaz
30008001	požadavek zápisu jednoho konfiguračního slova do CMD registru
0000000A	GRESTORE příkaz
20000000	prázdný příkaz
30008001	požadavek zápisu jednoho konfiguračního slova do CMD registru
00000005	START příkaz
20000000	prázdný příkaz
30008001	požadavek zápisu jednoho konfiguračního slova do CMD registru
00000007	příkaz pro reset CRC registru
30008001	požadavek zápisu jednoho konfiguračního slova do CMD registru
0000000D	příkaz DESYNCH
20000000	prázdný příkaz
20000000	prázdný příkaz



Obr. 40: Grafické znázornění adresy speciálního rámce

Maskovací rámce jsou součástí jednotlivých částečných konfiguračních souborů, přičemž každý tento soubor obsahuje maskovací rámce pro celý obvod mimo sebe sama – tj. při nahrání takového částečného souboru je možné příkazy vyčti (capture) a zapiš (restore) ovlivnit pouze tu část obvodu, pro kterou je určený daný konfigurační soubor).

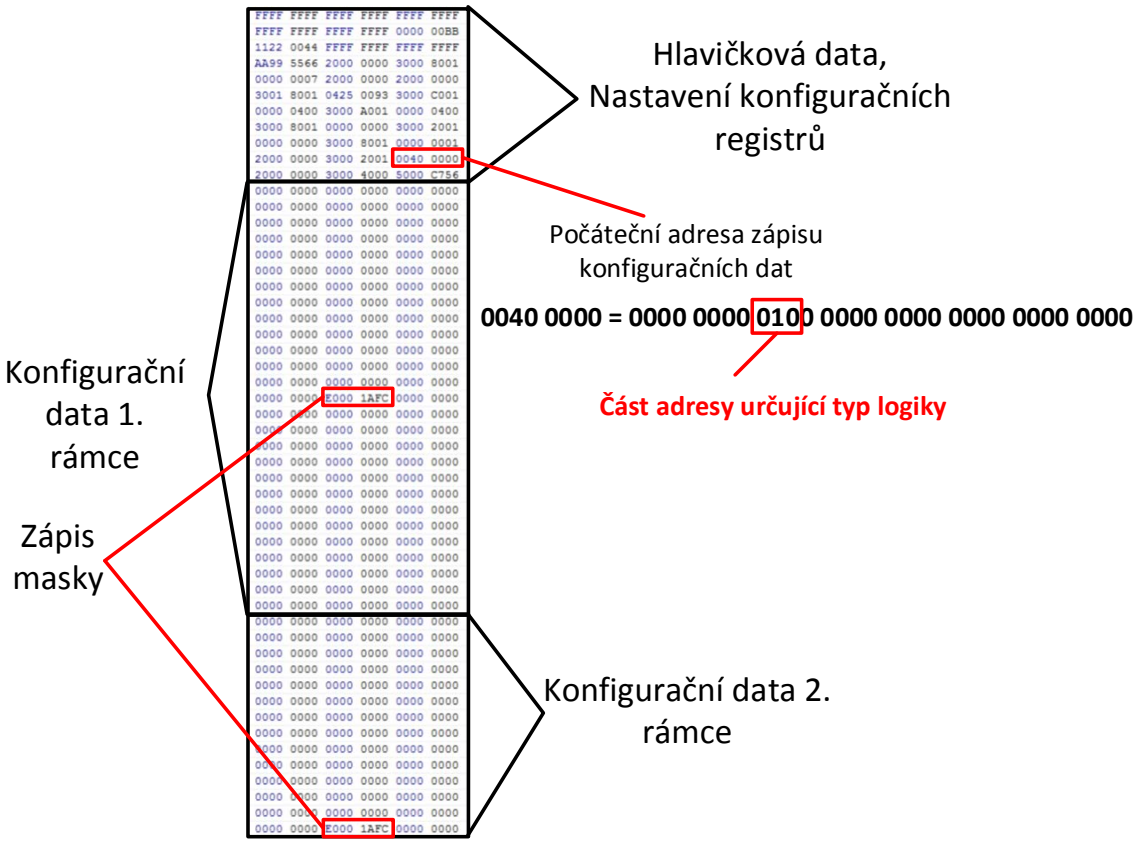
Použití maskování má tedy za následek zvětšení velikosti jednotlivých konfiguračních souborů (je třeba přidat zmíněné maskovací rámce), avšak proces maskování těch částí obvodu, které mají být chráněny, stačí provést pouze jednou při prvotní konfiguraci celého obvodu. Ukázku části konfiguračního souboru obsahujícího maskovací rámec můžeme vidět na Obr. 41. Tento rámec (stejně jako běžný rámec) obsahuje 81 konfiguračních slov, přičemž všechna konfigurační slova vyjma 41. jsou nulová. Adresování těchto maskovacích rámců začíná od adresy 0, stejně jako je tomu při adresování běžných konfiguračních rámců.

Vytvoření maskovacích rámců je možné provést manuálně (ručně nebo programově doplníme konfigurační soubor o potřebná maskovací data) nebo lze využít atribut RESET_AFTER_RECONFIGURATION zapisovaný do UCF souboru, který nastaví generátor konfiguračních souborů na vytvoření souborů obsahujících výše popsané maskovací rámce. Tato možnost je k dispozici v návrhovém prostředí PlanAhead od verze 14.3 (starší verze tento atribut nepodporují a při jeho zadání je ignorován). Tento atribut je možné využít pro obvody řady Virtex 6 a Virtex 7, pro starší obvody je nutné vytvořit maskovací rámce ručně. Kromě stručného popisu použití zmíněného atributu

není popis techniky pro zápis dat do interních registrů ani její implementace v dodávané dokumentaci uveden.

Příklad použití tohoto atributu v UCF souboru:

```
AREA_GROUP"nazev_rekonfig_oblati"RESET_AFTER_RECONFIG=TRUE;
```



Obr. 41: Výřez konfiguračního souboru – maskovací rámce

5 Experimentální výsledky

Pro účely testování bylo vytvořeno několik rekonfigurovatelných systémů. Při jejich návrhu byl kladen důraz na otestování správné funkce jednotlivých technik, které jsou v popisované metodice využívány, a následně byla otestována možnost jejich kombinace. Jednotlivé experimenty jsou seřazeny chronologicky s postupným vývojem této práce.

5.1 Relokace částečných konfiguračních souborů

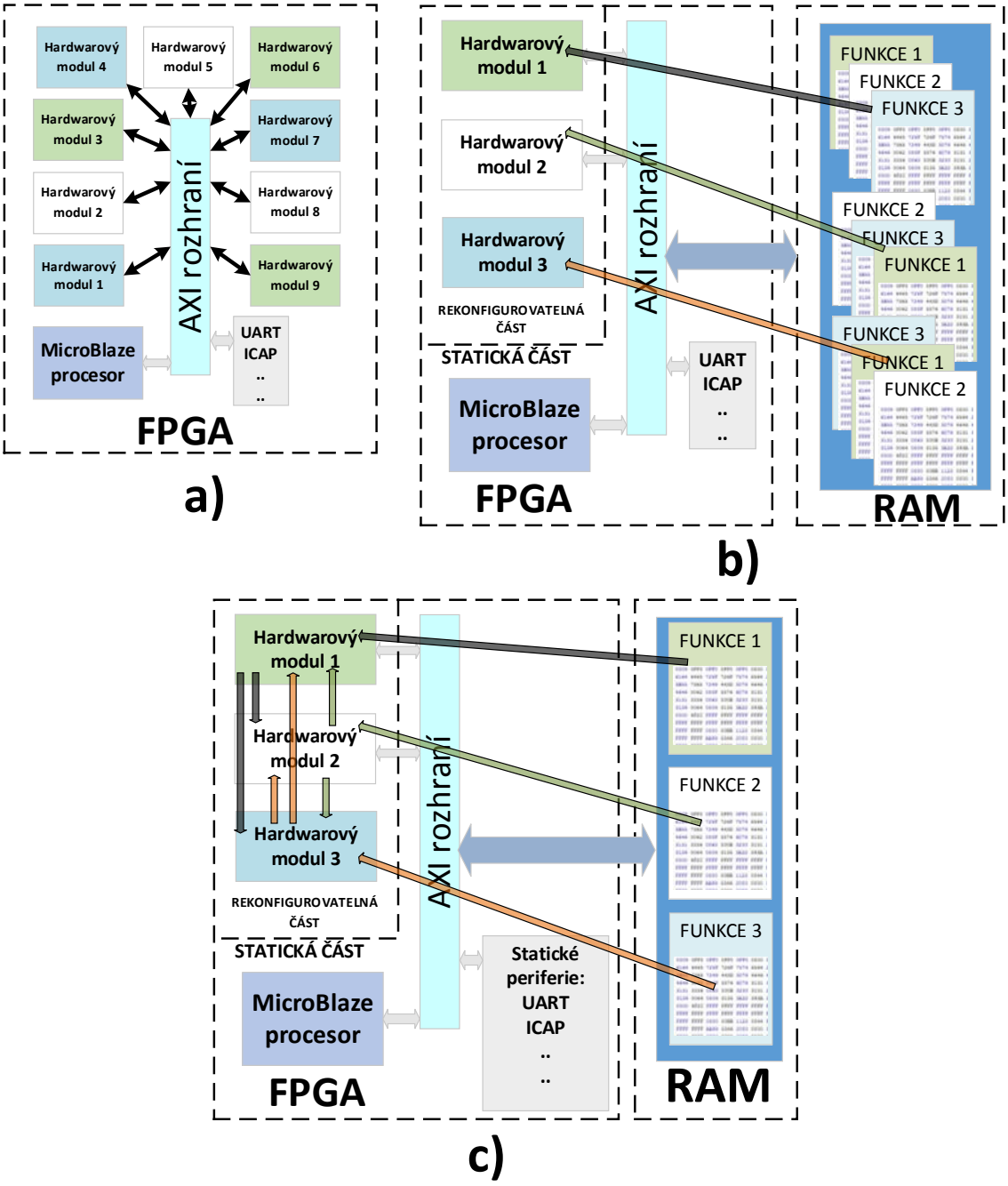
Postup pro vytvoření návrhu podporujícího techniku relokace částečných konfiguračních souborů je představen v kapitole 4.1. Tato technika je jedním ze stěžejních faktorů umožňujících dosažení vytyčených cílů práce.

Pro testování funkce a vlastností relokace byl vytvořen systém, na kterém lze demonstrovat výhody a nevýhody této techniky a odhadnout mez její použitelnosti. Mezí použitelnosti je myšleno určení typů systémů, pro které je tato technika vhodná z hlediska spotřeby logických zdrojů, potřebné paměti atd. Při tomto experimentu byl systém podporující relokaci srovnáván se systémem bez jakékoli podpory rekonfigurace a se systémem využívajícím standardní částečnou rekonfiguraci založenou na signálovém rozhraní tvořeném oddělovacími elementy. Všechny tři systémy mají stejnou funkčnost ve smyslu tří aktivních modulů ve stejný okamžik s tím, že tyto moduly mohou vykonávat různé funkce.

Sledovanými parametry je množství logiky potřebné pro vytvoření celého návrhu, množství paměti, které je nezbytné k uložení částečných konfiguračních souborů, a čas potřebný pro implementaci návrhu (doba od začátku implementace až po vytvoření konfiguračních souborů).

Množství potřebné logiky je závislé především na počtu hardwarových modulů, které jsou v systému využívány. Velikost paměti je úměrná počtu a velikosti rekonfigurovatelných modulů a množství funkcí, které mají tyto moduly vykovávat. Čas potřebný pro implementaci návrhu je závislý na celkové velikosti FPGA, na složitosti návrhu a na výpočetním výkonu, s jehož využitím je návrh zpracováván. S množstvím potřebné logiky ještě velmi úzce souvisí potřeba dodatečných logických zdrojů při použití rekonfigurace, která je přímo úměrná počtu vstupních a výstupních pinů jednotlivých modulů. Na každý pin je třeba umístit přemostění mezi statickou a dynamickou částí systému.

Na Obr. 42 jsou vyobrazena bloková schémata tří testovacích systémů. První představuje systém bez jakékoli podpory rekonfigurace (Obr. 42/a), druhý je systém s podporou standardní částečné rekonfigurace (Obr. 42/b) a třetí blokové schéma představuje systém s podporou relokace částečných konfiguračních souborů (Obr. 42/c).



Obr. 42: Blokové schéma testovací aplikace

V Tab. 4 je porovnáno celkové množství logiky (LUT elementy) potřebné pro vytvoření výše popsaných systémů s různým počtem pinů u použitých hardwarových modulů. Pro jednoduchost je velikost hardwarových modulů ve všech systémech stejná – 640 LUT elementů = dva rekonfigurovatelné rámce.

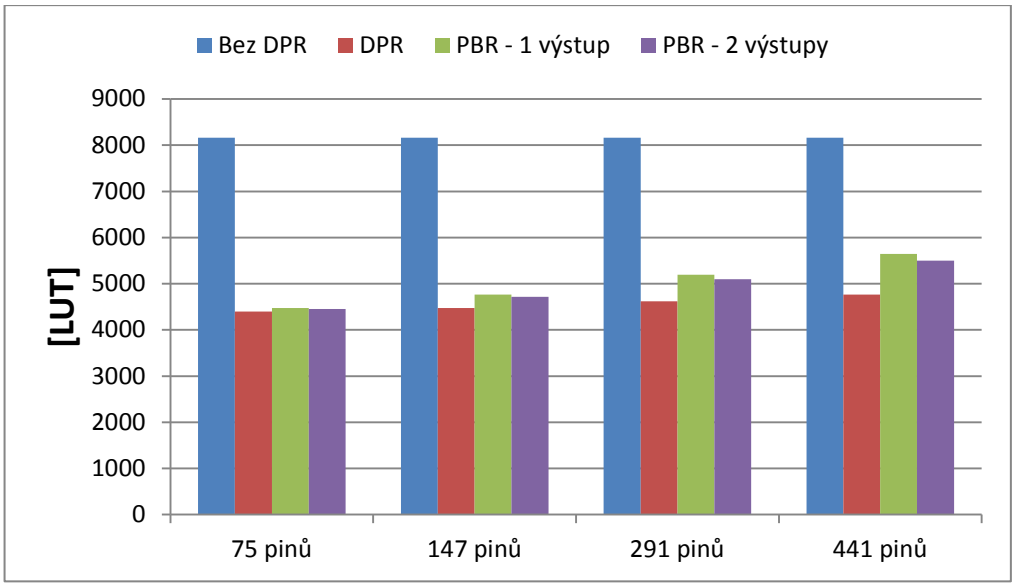
Tab. 4: Přehled potřebných LUT elementů pro jednotlivé typy systémů

Počet pinů – celkem/1 modul		**75/25**	**147/49**	**291/97**	**441/147**
Systém bez DPR	logika [LUT]	8165	8165	8165	8165
DPR	logika [LUT]	4400	4472	4616	4766
PBR – LUT jeden výstup	logika [LUT]	4475	4766	5198	5648
PBR – LUT oba výstupy	logika [LUT]	4450	4717	5101	5501

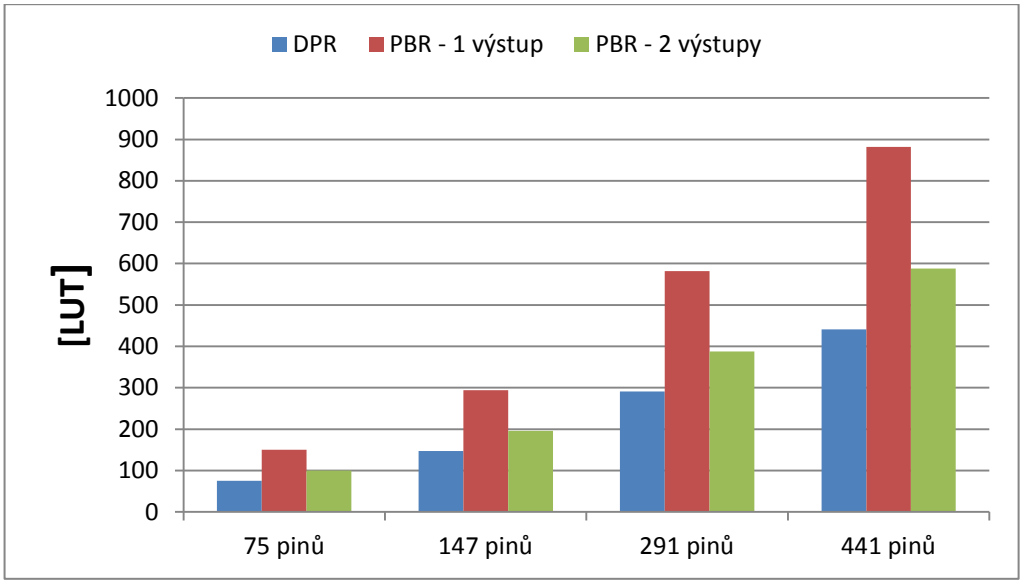
Celkové množství potřebné logiky je závislé na množství dodatečného hardwarového vybavení nezbytného pro vytvoření rekonfigurovatelného systému. Množství dodatečného hardwaru je shrnuto v Tab. 5. Počet pinů je uváděn jako množství pinů všech rekonfigurovatelných komponent v systému / množství pinů jedné komponenty. Počet pinů v prvních třech sloupcích odpovídá komponentám se dvěma n-bitovými vstupy, jedním n-bitovým výstupem a jedním řídicím signálem. Jedná se tedy o komponenty s osmi, šestnácti a 32bitovými vstupy a výstupy. V posledním sloupci je uvedeno množství pinů (147) potřebné pro připojení jedné komponenty na AXI rozhraní. Grafické znázornění celkem spotřebovaných LUT elementů je vyobrazeno na Obr. 43. Grafická závislost postupného navyšování potřebné logiky s narůstajícím počtem pinů je vyobrazena na Obr. 44.

Tab. 5: Přehled dodatečného hardwarového vybavení

Počet pinů – celkem/1 modul		**75/25**	**147/49**	**291/97**	**441/147**
Systém bez DPR	Režie [LUT]	0	0	0	0
DPR	Režie [LUT]	75	147	291	441
PBR – LUT jeden výstup	Režie [LUT]	150	294	582	882
PBR – LUT oba výstupy	Režie [LUT]	100	196	388	588



Obr. 43: Grafické znázornění celkové spotřeby logických zdrojů v obvodu



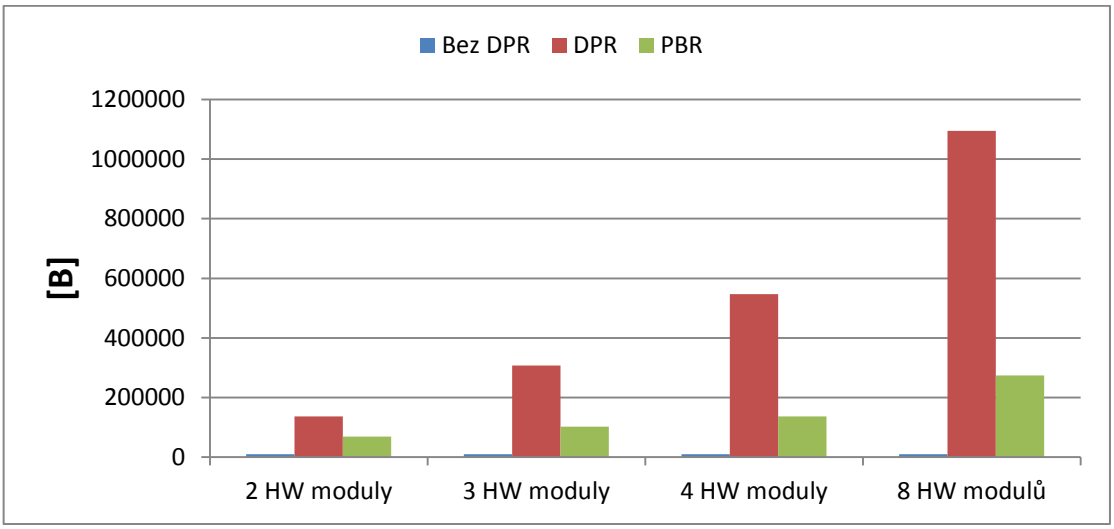
Obr. 44: Grafické znázornění navyšování dodatečných logických prvků v závislosti na počtu pinů u rekonfigurovatelných systémů

V následujících tabulkách je uveden přehled ostatních sledovaných parametrů. Uvedené hodnoty odpovídají systému, ve kterém dochází k rekonfiguraci, potažmo relokaci celých IP komponent připojených na rozhraní AXI (tj. jednotlivé moduly mají 147 pinů).

Tab. 6: Přehled potřebné paměti

Počet HW modulů		**2**	**3**	**4**	**8**
Systém bez DPR	paměť [B]	0	0	0	0
DPR	paměť [B]	136 800	307 800	547 200	1 094 400
PBR	paměť [B]	68 400	102 600	136 800	273 600

Tab. 6 shrnuje množství systémové paměti potřebné pro uložení částečných konfiguračních souborů v závislosti na počtu rekonfigurovatelných hardwarových modulů. Graficky je tato závislost zobrazena na Obr. 45.

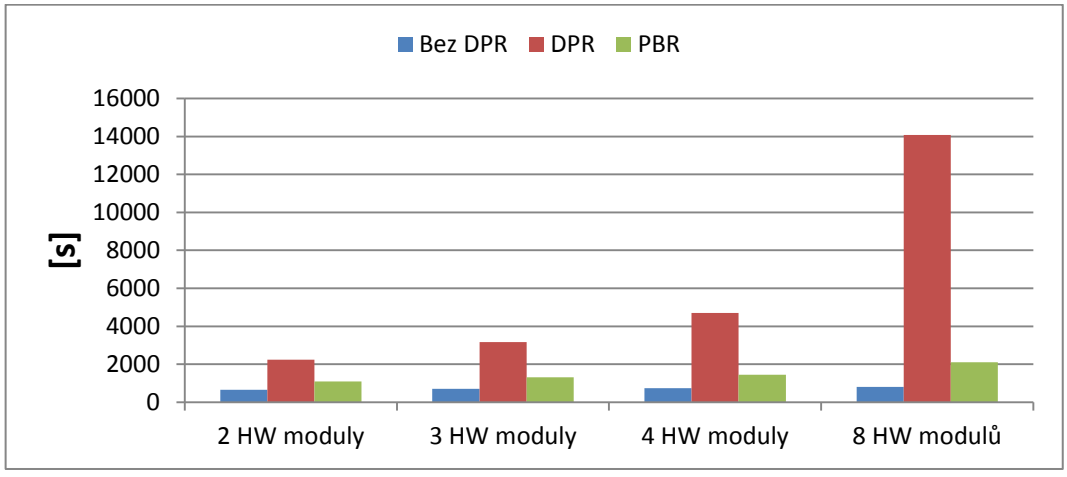


Obr. 45: Grafické znázornění nárůstu potřebné paměti bez a s použitím relokace

Posledním sledovaným parametrem je čas potřebný pro implementaci daného návrhu. Jak již bylo zmíněno výše, do tohoto času není započítána délka přípravy návrhu. Čas implementace zahrnuje pouze dobu běhu implementačních nástrojů. Časové nároky pro jednotlivé typy systémů jsou shrnuty v Tab. 7 a graficky znázorněny na Obr. 46.

Tab. 7: Přehled časových nároků implementace

Počet HW modulů		**2**	**3**	**4**	**8**
Systém bez DPR	čas [s]	657	702	736	806
DPR	čas [s]	2 246	3 165	4 694	14 072
PBR	čas [s]	1 099	1 305	1 439	2 102



Obr. 46: Grafický přehled času potřebného pro implementaci návrhu

5.1.1 Zhodnocení experimentu

Využití techniky rekonfigurace FPGA přináší snížení počtu logických zdrojů potřebných pro implementaci návrhu oproti systému, kde není rekonfigurace využita (viz Obr. 43).

Použití vytvořené metodiky pro relokaci částečných konfiguračních souborů mírně navyšuje počet potřebného hardwarového vybavení (Obr. 44), avšak oproti standardní částečné rekonfiguraci umožňuje vytvořená metodika značné snížení paměťových nároků (Obr. 45) a notně snižuje délku časového intervalu potřebného pro implementaci návrhu (Obr. 46).

Pro určení použitelnosti popsané metodiky pro relokaci částečných konfiguračních souborů je třeba zvážit, zda je tato technika pro daný konkrétní návrh výhodná či ne.

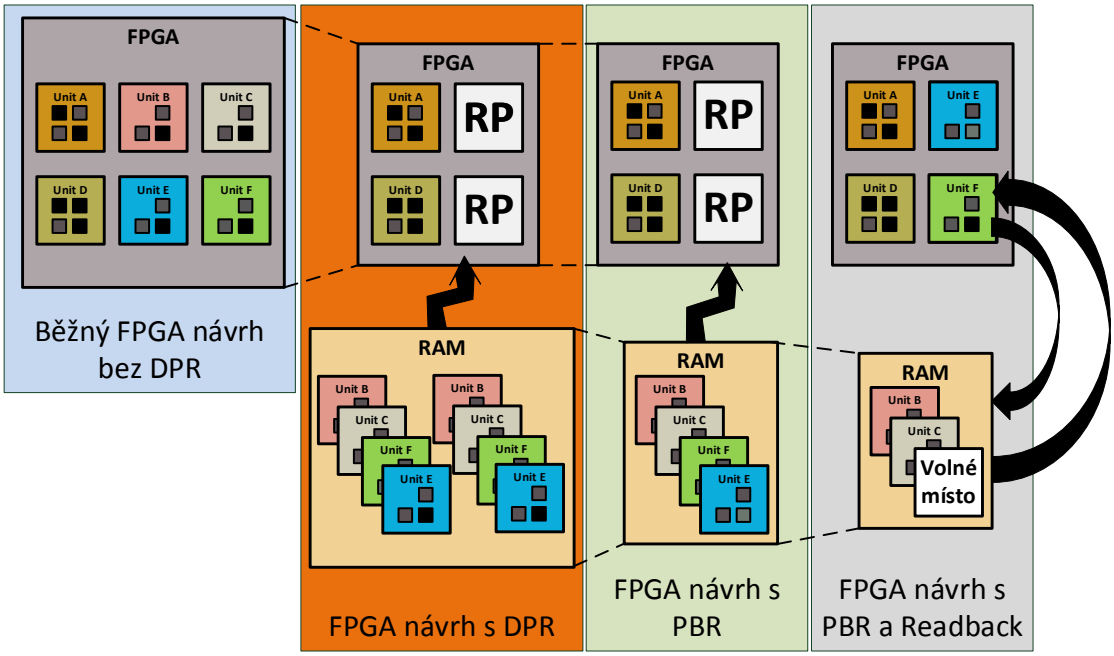
U návrhů s velkým množstvím malých rekonfigurovatelných oblastí a velkým počtem vstupních a výstupních pinů může použití této techniky způsobit nárůst dodatečného hardwarového vybavení přesahujícího svou velikostí množství logiky potřebné pro implementaci samotných hardwarových modulů. V takovýchto případech není použití relokace příliš vhodné vzhledem k tomu, že u malých modulů je množství ušetřené systémové paměti neúměrné k množství dodatečné logiky a ke zvýšené složitosti návrhu.

U návrhů, kde jsou rekonfigurovány velké hardwarové moduly, lze považovat dodatečné hardwarové vybavení za zanedbatelné a ostatní výhody relokace částečných konfiguračních souborů převáží zvýšenou složitost návrhu.

5.2 Zpětné vyčítání dat z konfigurační paměti

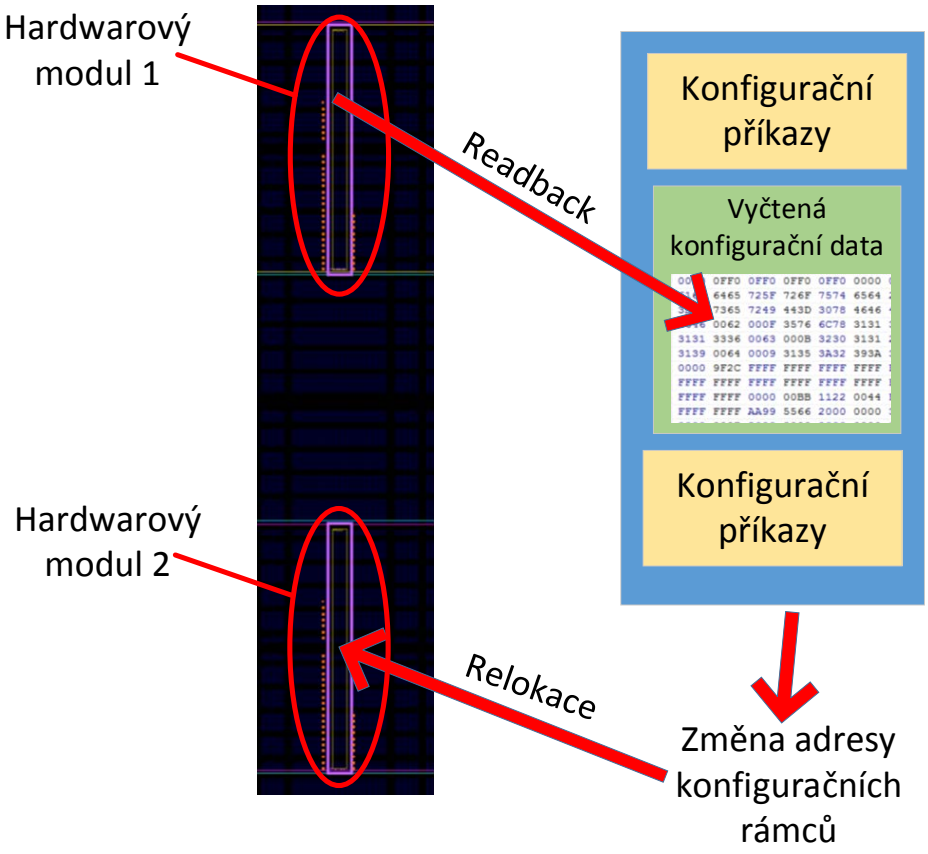
Techniku zpětného vyčítání konfigurační paměti lze použít i pro jiné účely, než pro které byla vytvořena (tj. ověření správnosti konfiguračních dat, funkční verifikace, odlaďování návrhu). Pokud zkombinujeme techniku relokace s technikou zpětného vyčítání, výsledná platforma může být použita pro snížení objemu konfiguračních dat, které je třeba uchovávat v systémové paměti. Navíc tato kombinace způsobuje další zvýšení flexibility systému, ve kterém je použita.

Moduly, které chceme relokovat (tj. zaměnit funkci jednoho modulu za funkci jiného modulu), jsou do FPGA nahrány již v rámci úplného konfiguračního souboru a pomocí techniky zpětného vyčítání konfigurační paměti můžeme získat všechna data popisující daný modul, změnit adresy konfiguračních rámců a takto upravená data nahrát na jiné místo v obvodu. Tato situace je principiálně naznačena na Obr. 47.



Obr. 47: Principiální schéma systému podporujícího techniku relokace a zpětného vyčítání konfigurační paměti

Z konfigurační paměti FPGA lze vyčíst data pro nastavení veškeré interní logiky a jejího propojení. Informace potřebné pro nastavení řídicích registrů obvodu v konfigurační paměti k dispozici nejsou. Je tedy nezbytné tyto řídicí příkazy uložit do systémové paměti spolu s adresami jednotlivých rekonfigurovatelných oblastí.



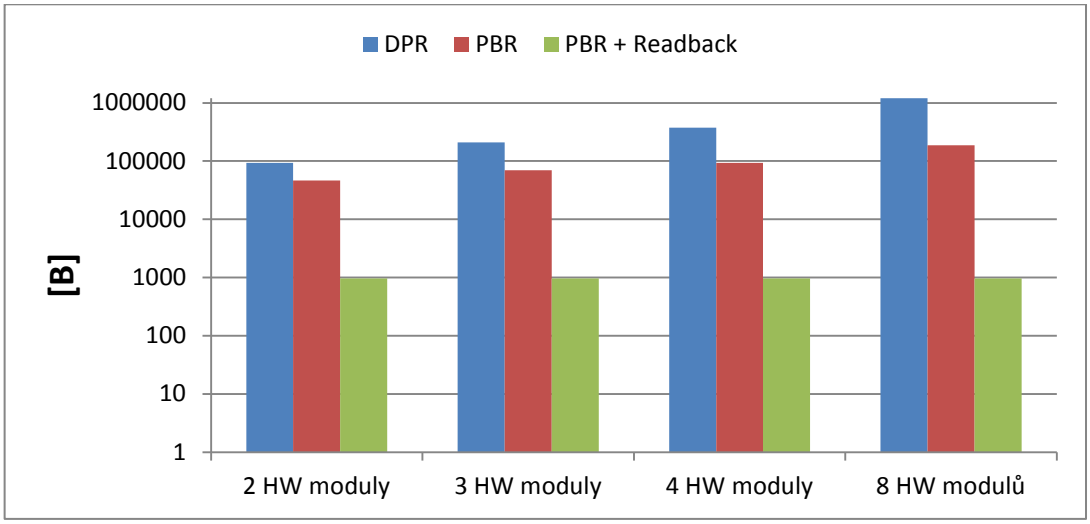
Obr. 48: Principiální schéma testovací aplikace získávající konfigurační data s využitím techniky zpětného vyčítání konfigurační paměti

Využití techniky zpětného vyčítání konfigurační paměti FPGA pro získávání konfiguračních dat bylo otestováno na systému obsahujícím dvě hardwarové komponenty. Pro jednoduchost testovací aplikace byly zvoleny hardwarové moduly zabírající oblast pouhých dvou rekonfigurovatelných rámců. Oba hardwarové moduly obsahují 32bitový čítač, přičemž první modul čítá vzestupně a druhý modul umožňuje čítání sestupné. Princip tohoto testovacího systému je naznačen na Obr. 48. Obě hardwarové komponenty jsou vytvořeny pomocí výše popsané metodiky pro relokaci. Tento fakt má za následek, že lze použít konfigurační data určená pro modul jedna i pro nastavení modulu dva. Popsaná aplikace umožňuje vzájemně přesouvat funkce mezi jednotlivými komponentami bez toho, aby bylo nutné uchovávat konfigurační data trvale v paměti.

Množství paměti potřebné pro trvalé uložení různého počtu hardwarových modulů je uvedeno v Tab. 8 a graficky znázorněno na Obr. 49. Systémová paměť je využita pouze v době, kdy se provádí samotná relokace (vyčtená konfigurační data musí být někde uložena). Trvale je uloženo pouze zanedbatelné množství příkazů pro řídicí registry.

Tab. 8: Přehled paměti potřebné pro trvalé uložení částečných konfiguračních souborů (vztaženo k HW modulům o velikosti dvou rekonfigurovatelných rámců)

Počet HW modulů		2	3	4	8
DPR	paměť [B]	93 312	209 952	373 248	1 492 992
PBR	paměť [B]	46 656	69984	93 312	186 624
PBR&Readback	paměť [B]	960	960	960	960



Obr. 49: Grafické znázornění paměti potřebné pro trvalé uložení částečných konfiguračních souborů

5.2.1 Zhodnocení experimentu

Kombinací techniky relokace částečných konfiguračních souborů s technikou zpětného vyčítání paměti FPGA je možné vytvořit vysoce flexibilní rekonfigurovatelný systém vyžadující minimální množství systémové paměti nutné pro ukládání konfiguračních dat.

Využitelnost výše popsané techniky je závislá na tom, jak rychle je třeba v dané aplikaci relokaci provést. Rychlost vyčítání a následné konfigurace je v tomto případě značně ovlivněna zvoleným přístupem k paměti. Přístup do systémové paměti je obsluhován samotným procesorem bez využití jakéhokoli DMA (Direct Memory Access) nebo jiné možnosti pro dávkové přenosy dat. Výsledná rychlost rekonfigurace (vyčtení z paměti a následný zápis) je tedy hluboko pod udávanou maximální rychlostí konfiguračního rozhraní ICAP.

Čas potřebný pro rekonfiguraci výše popsaným způsoben je uveden v Tab. 9. Pro zajímavost jsou zahrnuty i časy potřebné pro konfiguraci přes rozhraní JTAG a pro případ, kdy jsou konfigurační data uložena v paměti Compact Flash, jejíž rychlost je omezena maximální propustností hardwarové komponenty SysACE. Jednotlivé časové údaje opět odpovídají konfiguraci oblastí o velikosti dvou rekonfigurovatelných rámců.

Tab. 9: Porovnání časů potřebných pro konfiguraci

Způsob konfigurace	Compact Flash	JTAG	Vyčítání + konfigurace		
			Vyčítání	Konfigurace	Vyčítání + konfigurace
Přenosová rychlost [MB/s]	30	66	24	36	-
Množství přenesených dat [B]	186 624	186 624	178 944	186 624	365 568*
Čas potřebný pro konfiguraci [ms]	6,220	2,820	7,456	5,148	12,185

*množství přenesených dat je téměř dvojnásobné, před konfigurací je třeba data nejprve vyčíst (vyčtená data jsou kratší o nastavení CMD registrů)

Přenosové rychlosti u paměti Compact Flash a rozhraní JTAG jsou uvažovány maximální možné, které jsou uvedeny v oficiální dokumentaci výrobce. Rychlosti vyčítání a zápisu u prezentované techniky byly změřeny pomocí čítače umístěného v návrhu. Tyto rychlosti jsou přímo závislé na velikosti vyrovnávací paměti (paměť typu FIFO (First In First Out)) použité hardwarové komponenty HWICAP. Jedná se o standardní komponentu dostupnou v návrhových nástrojích Xilinx, která je určena pro obsluhu portu ICAP.

Pro dosažení lepší použitelnosti popsané kombinace technik je zapotřebí zvýšit maximální možnou rychlost vyčítání a zápisu dat do konfigurační paměti. A to ideálně až na hodnotu 400 MB/s, což je maximální rychlost ICAP portu udávaná výrobcem. Vyšší rychlosti rekonfigurace lze dosáhnout několika způsoby. Jednou z možností je doplnění návrhu o komponentu DMA, se kterou lze dosáhnout přenosové rychlosti až 70 MB/s [51]. Další možností je vytvoření vlastního řadiče ICAP portu – tento přístup je popsán např. v [50], kde autoři popisují vlastní komponentu pro správu rekonfigurace s možností dávkových přenosů konfiguračních dat. Uváděná rychlost téměř dosahuje výše avizovaných 400 MB/s. Popsaný přístup je výhodný především v tom, že veškerá agenda spojená s přenosem dat je v režii této komponenty a během rekonfigurace nedochází k nadměrnému zatěžování procesoru. Jiný přístup ke zvýšení rychlosti zápisu a čtení konfiguračních dat je popsán např. v [28]. Zde je využita standardní hardwarová komponenta HWICAP doplněná o vlastní logiku se schopností zápisu datových slov o vyšší bitové šíři a celá komponenta je přetaktována na nejvyšší možný kmitočet. Uváděná maximální přenosová rychlost je 2200 MB/s při přetaktování na frekvenci 550 MHz, což je více než pětinásobek výrobcem doporučené taktovací frekvence 100 MHz.

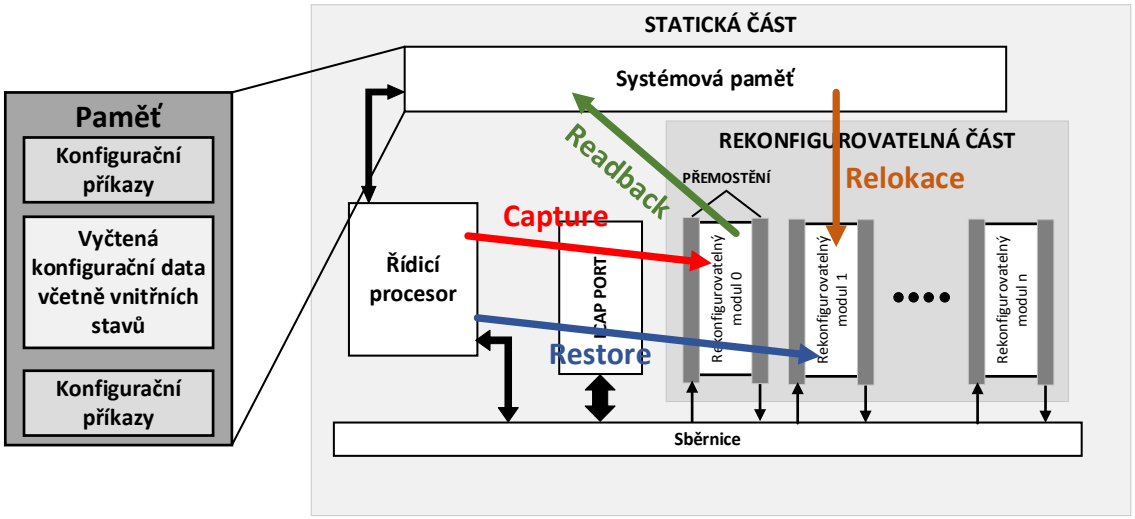
5.3 Zápis dat do interních registrů

Postup, jímž je možné přistupovat k technice zápisu dat do interních registrů FPGA, je popsaný v kapitole 4.3. Tato technika umožňuje zapsání hodnot do interních paměťových prvků na libovolné místo v obvodu. Daná oblast však nesmí být chráněna pomocí maskovacích rámců.

Nastavení příkazu pro zápis (restore) je v této práci realizováno zápisem posloupnosti příkazů do řídicích registrů FPGA. Vzhledem k nedostupnosti jakékoli oficiální dokumentace k technice zápisu dat do interních registrů byla sekvence potřebných příkazů (kapitola 4.3.1) sestavena experimentálně.

Využití techniky zápisu dat do interních registrů společně s technikami výše popsanými (tj. zkombinováním s technikou relokace a technikou zpětného vyčítání) umožňuje vytvoření rekonfigurovatelného systému podporujícího vzájemnou relokaci jednotlivých hardwarových modulů včetně jejich aktuálních vnitřních stavů. Při klasické DPR dochází k nastavení počátečních hodnot (reset) všech zainteresovaných komponent.

Před zahájením procesu přemisťování jednotlivých komponent je nejprve nastaven příkaz pro vyčtení hodnot vnitřních registrů (v okamžiku, kdy je žádoucí uložit vnitřní stavy). Poté jsou vyčtena všechna nezbytná data technikou zpětného vyčítání. Tato data jsou uložena do systémové paměti a následně jsou přidána k předuloženým řídicím příkazům, čímž dojde k vytvoření nového konfiguračního souboru. Dalším krokem je změna adresy a data jsou nahrána zpět (na jinou pozici) do systému s využitím ICAP portu. Posledním počinem před spuštěním nového modulu je zadání příkazu pro zápis dat do vnitřní logiky. Prvky obvodu, které nemají být při tomto procesu nijak ovlivněny, musí být chráněny pomocí maskovacích rámců. To znamená, že po vyčtení dat je třeba provést maskování zdrojové oblasti, aby nedošlo k jejímu ovlivnění při přepisování vnitřních stavů cílové oblasti. Blokové schéma výše popsaného testovacího systému je vyobrazeno na Obr. 50.



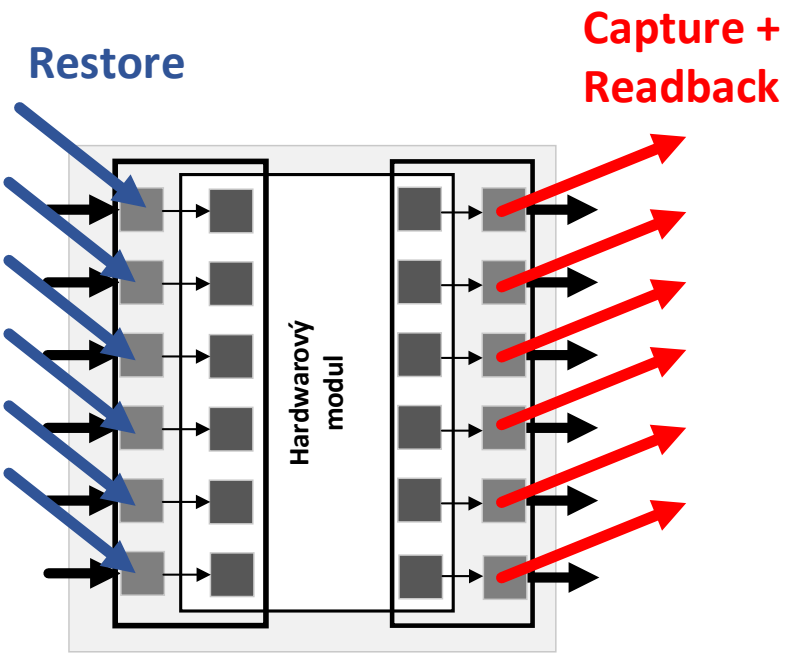
Obr. 50: Principiální schéma testovací aplikace umožňující relokaci modulů včetně jejich vnitřních stavů

5.3.1 Zhodnocení experimentu

Výše popsaná technika (respektive kombinace technik) umožňuje přemisťování funkčních modulů včetně jejich vnitřních stavů. Tento přístup je možné využít např. pro synchronizaci jednotlivých modulů v rekonfigurovatelných TMR systémech. Oproti jiným přístupům (kapitola 2) umožňuje tento způsob synchronizovat jednotlivé moduly bez nutnosti jejich fyzického spojení signálovými vodiči. Tato vlastnost přispívá k zjednodušení propojení celého systému.

Uspořádání vstupních a výstupních oddělovacích maker vytvořených z důvodu podpory relokace komponent je příhodné pro eventuální využití při testování těchto komponent. Techniku zápisu dat do interních registrů je možné využít pro vkládání testovacích dat. Reakce testovaného obvodu na tato data je možné na výstupech sledovat pomocí techniky zpětného vyčítání. Princip takovéhoto způsobu využití je vyobrazen na Obr. 51.

Technika zápisu dat do interních registrů může být použita i samostatně. Je možné vkládat data na libovolné nemaskované místo v obvodu, to znamená, že daná oblast nemusí být rekonfigurovatelná. Jelikož nejmenší oblast obvodu, kterou lze samostatně maskovat (odmaskovat), je oblast jednoho sloupce logiky o výšce hodinového řádku, je třeba vždy počítat s tím, že po zadání příkazu pro zápis dat dojde k přepsání všech hodnot v daném rekonfigurovatelném rámci.



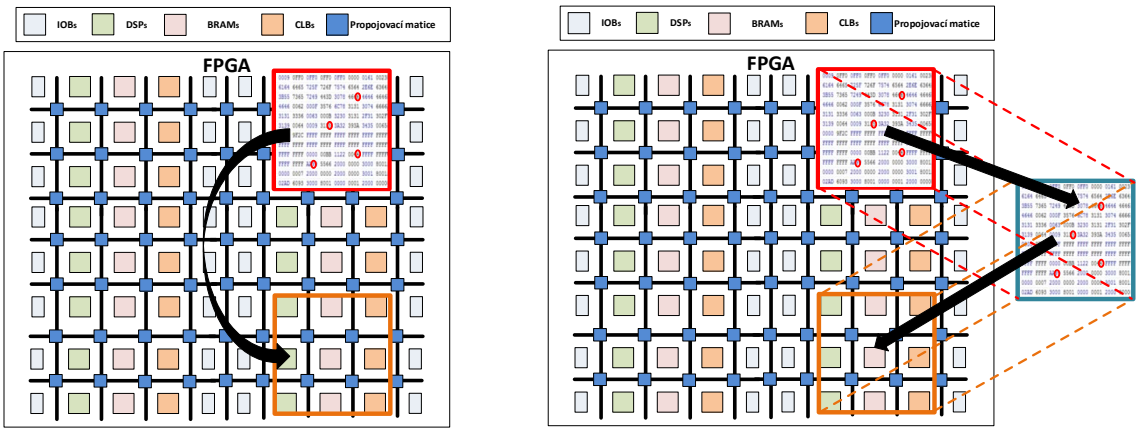
Obr. 51: Princip využití struktury oddělovacích maker pro testování hardwarového modulu

Výsledkem kombinace výše popsaných technik je procedura, kterou lze obecně pojmenovat jako relokace hardwarových úloh. Pro přenos vnitřních stavů mezi jednotlivými moduly lze použít více přístupů s různými nároky na složitost, rychlost konfigurace a spotřebu paměti v daném systému.

Jedním z nich je přístup použitý v předcházejících experimentech (kapitoly 5.2 a 5.3), kdy jsou data potřebná pro relokaci kompletně vyčtena z konfigurační paměti obvodu, kromě řídicích příkazů, které nejsou součástí konfiguračních dat a jsou předpřipravené v systémové paměti. Po vyčtení dojde ke změně adresy prvního konfiguračního rámce (ostatní adresy jsou inkrementovány automaticky) a takto vytvořený částečný konfigurační soubor je zapsán na jiné místo v obvodu. Tato situace je blokově naznačena v levé části na Obr. 52. Při použití tohoto přístupu je nutné počítat s tím, že doba potřebná pro nahrání konfiguračního souboru je ovlivněna téměř dvojnásobným množstvím dat, která musejí být přenesena.

Druhý přístup pro relokaci hardwarových úloh je založen na tom, že z daného hardwarového modulu jsou vyčteny pouze konfigurační rámce se stavy vnitřních registrů. Tyto hodnoty jsou následně zapsány do souboru předem uloženého v systémové paměti. Tento soubor je poté nahrán na určené místo v obvodu. Popsaný přístup je blokově naznačen v pravé části na Obr. 52. Oproti předchozímu případu vyžaduje vyčítání menšího množství dat. Ke zjištění polohy konkrétních konfiguračních

rámců s požadovanými daty je nezbytné využít informací uložených v alokačním souboru konfigurační paměti (viz kapitola 4.2.1).



Obr. 52: Různé přístupy relokace hardwarových úloh

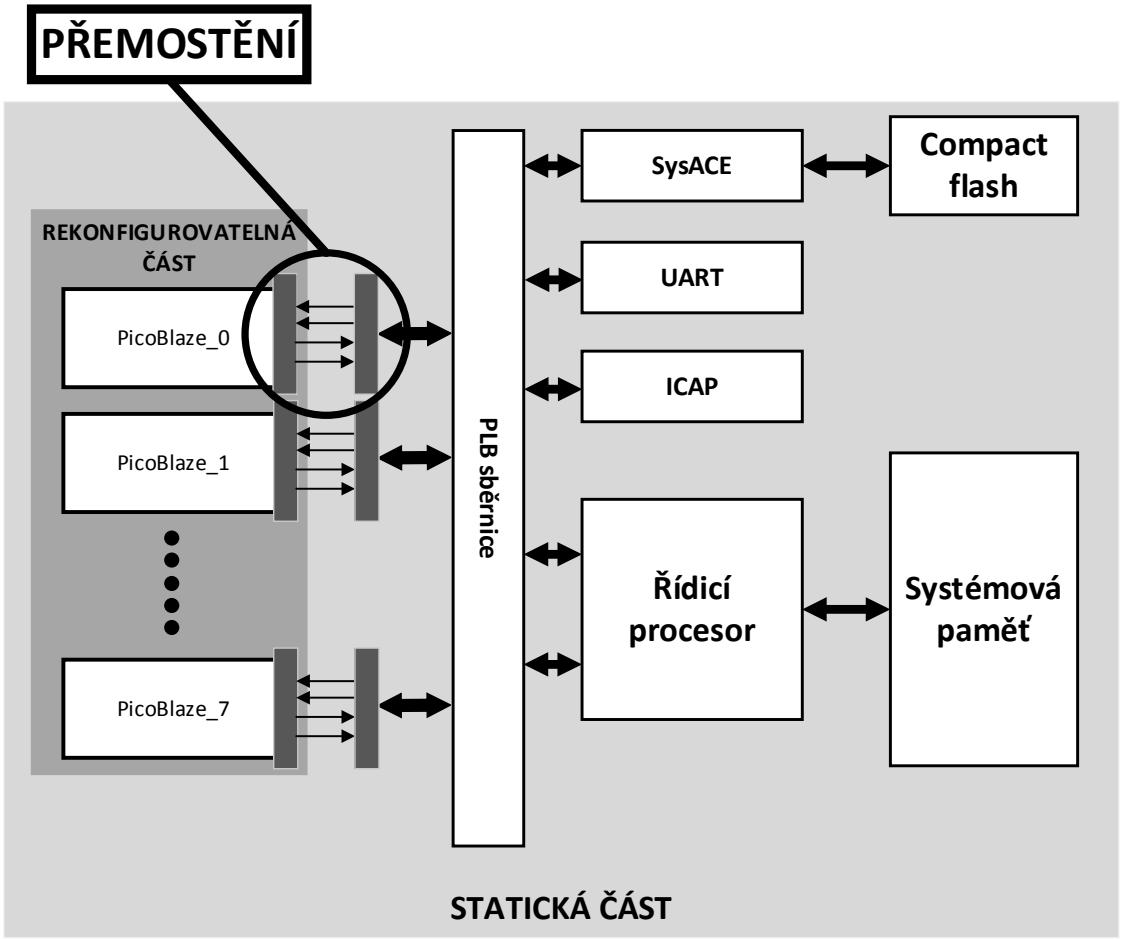
5.4 Test na komplexním systému

Všechny výše popsané experimenty jsou založeny na využití hardwarových modulů o velikosti dvou rekonfigurovatelných rámců, avšak funkční bloky umístěné uvnitř těchto modulů obsahují pouze jednoduché komponenty (násobička, sčítačka, čítač), které jsou tvořeny z malého počtu logických prvků. V některých případech lze dokonce říci, že navýšení potřebné logiky pro implementaci daného modulu způsobené přemostěním statické a dynamické části systému je vyšší než množství logiky použité pro implementaci samotných hardwarových modulů. Série těchto jednoduchých experimentů byla nezbytná pro ověření funkce dílčích částí práce.

Pro demonstraci použitelnosti popisované metodiky byla kombinace popsaných technik odzkoušena na komplexnějším systému. Základem tohoto systému je statický řídicí procesor MicroBlaze s potřebnými periferiemi a osmice rekonfigurovatelných procesorů PicoBlaze. PicoBlaze představuje kompaktní vestavné osmibitové procesorové jádro s redukovanou instrukční sadou (RISC – Reduced Instruction Set Computing). Tento procesor je optimalizovaný pro implementaci do obvodů od společnosti Xilinx [63].

Popisovaný testovací systém obsahuje ve své statické části procesor MicroBlaze, dále je zde systémová paměť, komunikační rozhraní UART, konfigurační port ICAP a řadič paměti Compact Flash (externí úložiště částečných konfiguračních souborů). Dynamická část tohoto systému je tvořena osmi rekonfigurovatelnými oblastmi pro jednotlivé procesory PicoBlaze. K propojení všech komponent v systému je využita

sběrnice PLB (Processor Local Bus). Blokové schéma tohoto systému je vyobrazeno na Obr. 53.



Obr. 53: Blokové schéma testovací aplikace s procesory PicoBlaze

Tato aplikace umožňuje relokaci jednotlivých jader procesoru PicoBlaze včetně přenosu aktuálního vnitřního stavu každého procesoru. Program pro procesor PicoBlaze napsaný pomocí jazyka symbolických adres (assembleru) je převeden do jazyka VHDL a následně implementován spolu s ostatní logikou procesoru. Vzhledem k tomu je pro změnu programu vždy třeba re-implementovat jádro procesoru. Skutečné rozmístění rekonfigurovatelných oblastí je zobrazeno na Obr. 61 v příloze B.

Využití techniky relokace umožňuje vytvořit pro každou požadovanou funkci procesoru jeden konfigurační soubor, který je možné použít pro všechny hardwarové moduly. Velikost oblastí zabíraná jednotlivými moduly odpovídá šesti rekonfigurovatelným rámcům. Z toho jsou čtyři rámce logiky (SLICEs) a dva rámce pro nastavení paměti BRAM (jeden pro nastavení vlastností paměti a jeden pro její obsah). Velikost paměti potřebná pro trvalé uložení konfiguračních souborů, délka času

potřebná pro implementaci návrhu a délka času potřebná pro nahrání některého z modulů jsou shrnuty v následujících tabulkách.

Tab. 10: Přehled množství paměti potřebné pro uložení částečných konfiguračních souborů u systémů s různým počtem rekonfigurovatelných procesorů PicoBlaze

Počet modulů		2	4	8
DPR	**Paměť[B]**	202188	808752	3235008
PBR	**Paměť[B]**	101094	202188	404376
PBR&Readback	**Paměť[B]**	960	960	960

Tab. 11: Přehled času potřebného pro implementaci systémů s různým počtem rekonfigurovatelných procesorů PicoBlaze

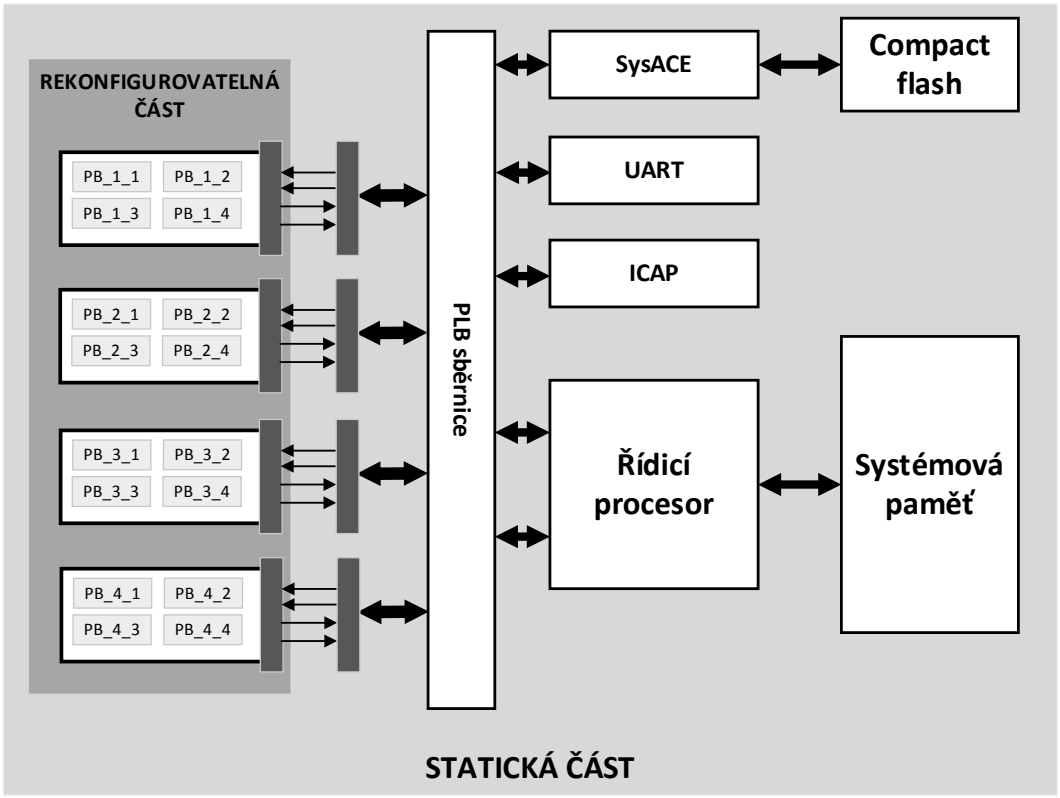
Počet modulů		2	4	8
DPR	**Čas [s]**	3158	21064	82456
PBR	**Čas [s]**	2699	3078	3630
PBR&Readback	**Čas [s]**	2730	3135	3726

Tab. 12: Přehled času potřebného pro rekonfiguraci jednoho modulu s procesorem PicoBlaze

Konfigurace přes ICAP port	**Vyčítání + konfigurace**		
	Vyčítání	**Konfigurace**	**Celkem**
Přenosová rychlost [MB/s]	24	36	-
Množství přenesených dat [B]	56587	57547	114134
Čas potřebný pro rekonfiguraci [ms]	18,862	12,788	31,650

Výška jednotlivých rekonfigurovatelných oblastí musí být vždy rovna minimálně výšce jednoho CLK řádku (viz kapitola 1.2). Implementované procesory PicoBlaze využívají pouze jeden blok paměti BRAM, to znamená, že jejich umístění do jednotlivých rekonfigurovatelných oblastí (viz příloha B Obr. 61) vykazuje značnou neefektivitu využití logických zdrojů obvodu.

Tento nedostatek byl odstraněn mírnou úpravou popsaného systému, kdy namísto osmice rekonfigurovatelných jednotlivých procesorů PicoBlaze nová platforma obsahuje čtyři rekonfigurovatelné komponenty, přičemž každá tato komponenta je složená ze čtyř nezávislých jader procesoru PicoBlaze. Tento modifikovaný systém je blokově naznačen na Obr. 54 a pohled na skutečné uspořádání uvnitř obvodu zobrazen na Obr. 62 v příloze C. Implementační detaily tohoto testovacího systému jsou shrnuty v Tab. 13 a graficky znázorněny na Obr. 54, Obr. 55 a Obr. 56.

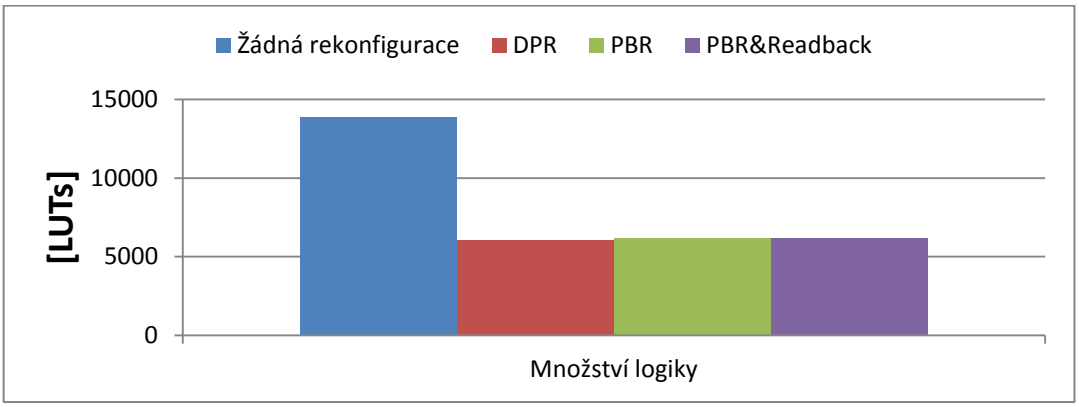


Obr. 54: Blokové schéma testovací modifikované aplikace s procesory PicoBlaze

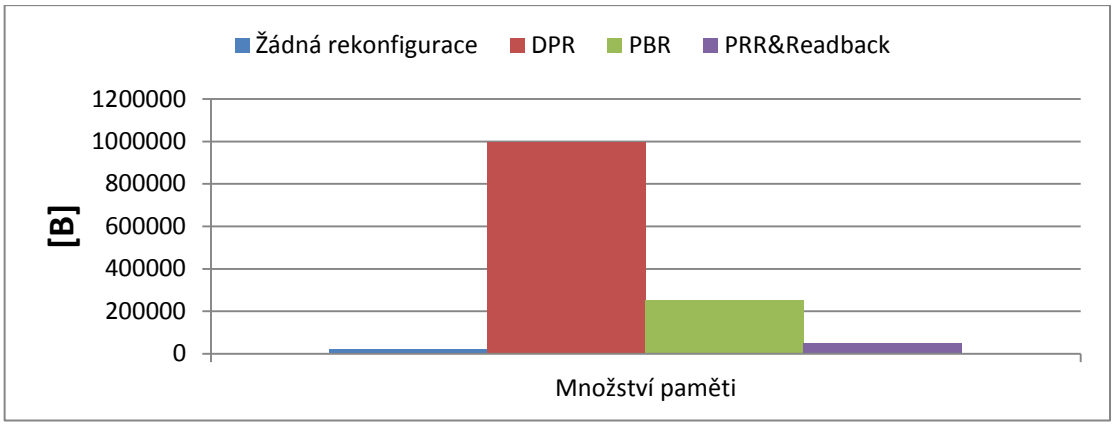
Tab. 13: Přehled základních parametrů testovacího systému

	Logika [LUT]	Paměť [B]	Implementační Čas [s]	Čas rekonfigurace [ms]
Žádná rekonfigurace	13 888	0	830	-
DPR	6 064	997 680	28 640	13,856
PBR	6 192	249 420	1 790	13,856*
PBR&Readback	6 192	960	1 910	34,321

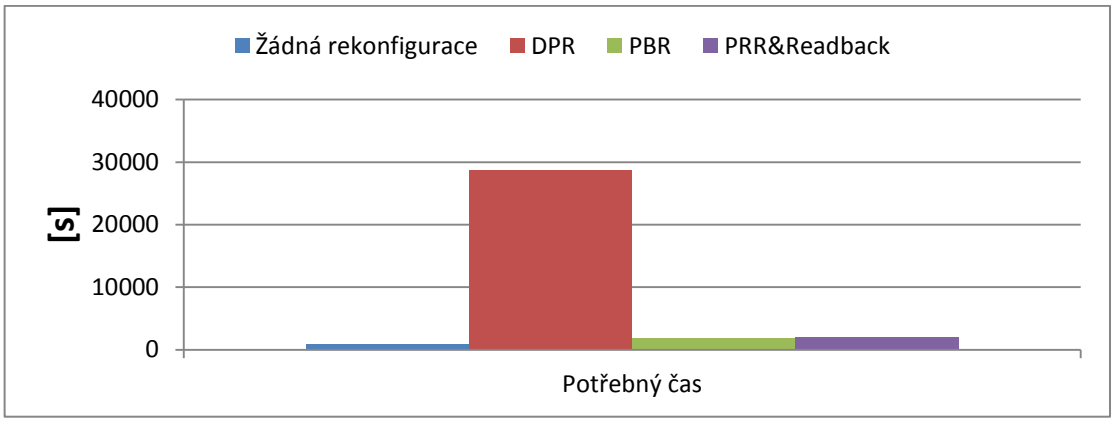
* Do doby potřebné pro provedení rekonfigurace není započítána doba potřebná pro změnu adresy rámce



Obr. 55: Graficky znázorněné množství logiky nutné pro implementaci testovacího systému se čtyřnásobným procesorem PicoBlaze



Obr. 56: Graficky znázorněné množství paměti potřebné pro trvalé uložení konfiguračních souborů u testovacího systému se čtyřnásobným procesorem PicoBlaze



Obr. 57: Graficky znázorněné množství času potřebného pro běh implementačních nástrojů u testovacího systému se čtyřnásobným procesorem PicoBlaze

5.4.1 Zhodnocení experimentu

Popsané experimenty ukazují, že metodika návrhu prezentovaná v této práci je použitelná i na komplexní systémy, kde dochází k rekonfiguraci poměrně složitých hardwarových modulů. V Tab. 10 a Tab. 11 jsou porovnány paměťové a časové nároky systému s procesory PicoBlaze (viz Obr. 53), kde bylo do systému postupně přidáváno větší množství hardwarových modulů. V tabulkách jsou porovnány tři typy rekonfigurovatelných systémů, přičemž první umožňuje pouze podporu standardní DPR, druhý podporuje relokaci funkčních modulů a třetí typ je systém s podporou relokace jednotlivých komponent včetně jejich aktuálních vnitřních stavů. Tab. 12 shrnuje časy potřebné pro rekonfiguraci jednoho hardwarového modulu s využitím metodiky založené na kombinaci technik relokace, zpětného vyčítání a zápisu dat do vnitřní logiky.

Obdobou tohoto experimentu je testovací aplikace blokově vyobrazená na Obr. 54, kde došlo ke zvýšení hustoty využití logických zdrojů v jednotlivých

rekonfigurovatelných oblastech. Tento fakt vedl mimo jiné ke zvýšení počtu vstupních a výstupních pinů jednotlivých modulů, a tím i ke zvýšení složitosti při mapování a propojování návrhu. Provedený experiment dokládá, že popsaná metodika je použitelná i pro velké funkční moduly (osm rekonfigurovatelných rámců). Časování celého systému bez problému splňuje požadovaných 100 MHz (výsledky implementace uvádějí maximální taktovací frekvenci 130,856 MHz). V Tab. 13 jsou shrnuty základní parametry této platformy, mezi které patří množství potřebné logiky, množství potřebné paměti, čas potřebný pro implementaci a čas potřebný pro rekonfiguraci jednoho hardwarového modulu. První tři z těchto parametrů jsou graficky znázorněny na Obr. 55, Obr. 56 a Obr. 57. V porovnání jsou zahrnuty čtyři typy systémů, prvním je systém bez podpory částečné rekonfigurace, další tři jsou systémy s podporou různých typů rekonfigurace (běžná DPR, PBR, PBR včetně vnitřních stavů hardwarových komponent). Tyto testovací systémy jsou navrženy tak, aby umožňovaly nastavení stejného množství funkcí jednotlivých hardwarových modulů. To znamená, že pokud má daná rekonfigurovatelná platforma čtyři hardwarové moduly, je v případě statického systému zapotřebí vytvořit šestnáct hardwarových modulů, aby bylo dosaženo stejné variability funkcí. U běžné DPR je třeba pro každý z těchto hardwarových modulů vytvořit a uchovávat čtyři konfigurační soubory.

Závěr

Prezentovaná metodika návrhu je založena na použití několika dílčích technik, které společně umožňují vytvoření komplexního vysoce flexibilního systému na FPGA obvodu. V následujícím textu jsou shrnuty vlastnosti těchto dílčích technik a je zde zhodnocen přínos jejich použití.

Se stále se zlepšující podporou částečné rekonfigurace v návrhových nástrojích se tato technika objevuje v celé řadě aplikací (kapitola 3). Naneštěstí její použití vykazuje některé aspekty (složitost návrhu, dodatečné navýšení potřebného implementačního času, paměti a logických prvků v obvodu), díky kterým je v řadě případů nevýhodná. Tato práce popisuje metodiku návrhu rekonfigurovatelných systémů na FPGA obvodech Xilinx. Prezentovaný postup umožňuje vytvoření optimalizovaného systému z hlediska výše zmíněných negativních parametrů.

Prvním dílčím počinem v rámci této práce bylo zpracování metody pro vytvoření návrhu s podporou relokace částečných konfiguračních souborů. Metodika využívá běžných implementačních nástrojů, tj. Xilinx Design Tools se zakoupenou licencí pro částečnou dynamickou rekonfiguraci. Principiálně je tato metoda založena na transformaci oddělovacích elementů na útvar, který lze definovat jako oddělovací makro.

Standardní oddělovací makro se používalo u starších FPGA obvodů (Virtex 4) a bylo podporováno v nižších verzích návrhových nástrojů (např. Xilinx ISE 9.2). Pro toto makro bylo nutné nejprve definovat správný typ a poté ho manuálně vložit do návrhu. U vytvořené metody je každé oddělovací makro tvořeno z oddělovacího elementu a z jednoho dodatečně přidaného LUT elementu. Volba směru a způsobu propojení těchto LUT elementů je zcela v rukou automatických propojovacích nástrojů. Tento postup vytváření rekonfigurovatelného systému sice navyšuje potřebu dodatečného hardwarového vybavení oproti přístupům využívajícím standardní oddělovací elementy. Avšak při vhodné volbě velikosti hardwarových modulů a množství pinů je toto navýšení potřebné logiky zastíněno ostatními vlastnostmi relokace částečných konfiguračních souborů. Navýšení složitosti návrhu, které je způsobeno nutností manuálně vytvářet jednotlivá oddělovací makra, je kompenzováno automatizací tohoto procesu s využitím sady vytvořených skriptů. Tyto skripty jsou společně s elektronickou verzí tohoto textu umístěny na přiloženém CD.

Po rekonfiguraci je třeba nově umístěnou jednotku inicializovat, případně synchronizovat se zbytkem systému. Některé typy FPGA podporují techniku zpětného vyčítání konfigurační paměti (readback), díky které lze z obvodu získat veškerá konfigurační data i s obsahem vnitřních registrů v dané části obvodu. Dále pak bývá podporována technika zápisu dat do interních registrů obvodu (restore), která umožňuje inicializaci vnitřních stavů interních registrů v obvodu. Tyto techniky lze použít pro synchronizaci nových prvků v systému bez nutnosti jejich fyzického propojení. To může vést ke značné úspoře dodatečného hardwarového vybavení oproti systémům, které podporují synchronizaci s využitím datového propojení jednotlivých modulů.

Metodika návrhu prezentovaná v této práci inovativně umožňuje provádět relokaci hardwarových komponent na FPGA obvodech řady Virtex 5 a vyšší. Při práci s těmito obvody je přemostění mezi statickou a dynamickou částí řešeno pomocí oddělovacích elementů, které za normálních okolností relokaci neumožňují. Přístup vytvořený v rámci této práce nevyžaduje provedení žádných významných zásahů do konfiguračních souborů nebo do strukturních netlistů, jako je tomu např. u nástroje GoAhead, kde dochází k jejich úpravě pomocí jazyka XDL.

Dalším inovativním prvkem je využití částečné rekonfigurace spolu s pokročilými technikami, mezi něž patří relokace hardwarových modulů, zpětné vyčítání konfigurační paměti a zápis dat do interních registrů obvodu. Tato kombinace umožňuje vytvořit systém podporující dynamické přemísťování hardwarových komponent včetně jejich aktuálních stavů. Dále je možné provádět synchronizaci funkčních komponent po jejich rekonfiguraci. Případně je možné měnit obsah libovolného paměťového prvku v obvodu.

Způsob tvorby rekonfigurovatelných systémů prezentovaný v předcházejícím textu celistvě popisuje možnosti využití pokročilých vlastností moderních FPGA obvodů s možností tvorby vysoce flexibilního návrhu splňujícího ty nejvyšší nároky.

Seznam literatury

Vlastní publikace

IEEE publikace:

[1] Drahoňovský, T.;Rozkovec, M.;Novák, O.: "A highly flexible reconfigurable system on a Xilinx FPGA," International Concerence on ReConFigurable Computing and FPGAs (ReConFig 2014), 8-10 December 2014, ISBN 978-1-4799-5943-3

[2] Drahoňovský, T.;Rozkovec, M.;Novák, O.: "Relocation of reconfigurable modules on Xilinx FPGA," Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2013 IEEE 16th International Symposium., pp.175 - 180, 8-10 April 2013, IEEE Catalog Number: CFP13DDE-PRT, ISBN 978-1-4673-6133-0

[3] Cvek, P.;Drahoňovský, T.; Rozkovec, M.: "GNU/Linux and Reconfigurable Multiprocessor FPGA Platform," Electronics, Control, Measurement, Signals and their application in Mechatronics (ECMSM2013), Toulouse, France, June 2013, IEEE Catalog Number: CFP13ECN-USB, ISBN: 978-1-14673-6297-9

Ostatní publikace:

[4] Drahoňovský, T.;Rozkovec, M.: "Guitar effects implementation in the FPGA circuits," Electronics, Control, Measurement and Signals (ECMS), June 2011, Liberec, Czech republic, Informal proceedings of the ECMS 2011, pp. 80-86, ISBN 978-80-7372-781-9

[5] Drahoňovský, T.: " Implementace kytarových efektů v obvodu FPGA," 2011, Sdělovací technika, No. 2, Febuary 2011, pp. 4-7, ISSN 0036-9942

[6] Drahoňovský, T.: "Návrh rekonfigurovatelného víceprocesorového systému na FPGA obvodu," 2012, Počítačové architektury & diagnostika (PAD), September 2012, Milovy, Czech republic, pp. 85-90, ISBN 978-80-01-05106-1

[7] Drahoňovský, T.: "Využití částečné dynamické rekonfigurace pro testování a zvyšování spolehlivosti FPGA obvodů," 2011, Počítačové architektury & diagnostika (PAD), September 2011, Stará Lesná, Slovakia, pp. 55-60, ISBN 978-80-227-3552-0

[8] Drahoňovský, T.: "Hardware task relocation on a Xilinx FPGA," 2014, Annual DCPS Evaluation Workshop, November 2014, Cottbus, Germany, pp. 35-38

[9] Drahoňovský, T.: “Reconfigurable system on Xilinx FPGA with low memory requirements for partial bitstreams storing,” 2014, 3rd Biannual European – Latin American Summer School on Design, Test and Reliability, April 2014, Frankfurt (Oder), Germany, pp. 115-119

[10] Drahoňovský, T.: “Víceprocesorový rekonfigurovatelný systém na obvodu FPGA,” Počítačové architektury & diagnostika (PAD), September 2013,Klášter Teplá, Czech republic, pp. 75-80, ISBN 978-80-261-0270-0

Použitá literatura

[11] Abramovici, M.; et al.: “Using roving STARs for on-line testing and diagnosis of FPGAs in fault-tolerant applications,” in Int’l Test Conference, 1999, pp. 973–982.

[12] Anjam, F.; Nadeem, M.; Wong, S.: “A VLIW softcore processor with dynamically adjustable issue-slots,” Field-Programmable Technology (FPT), 2010 International Conference on , vol., no., pp.393,398, 8-10 Dec. 2010

[13] Bauer, L.; Braun, C.; Imhof, M.E.; Kochte, M.A; Hongyan Zhang; Wunderlich, H.; Henkel, J.: “OTERA: Online test strategies for reliable reconfigurable architectures,” Adaptive Hardware and Systems (AHS), 2012 NASA/ESA, pp.38,45, 25-28 June 2012

[14] Becker, P. C. T.; Luk, W.: “Enhancing relocatability of partial bitstreams for run time reconfiguration,” in Proc. 15th Annu. IEEE Symp. Field-Program. Custom Comput. Mach., Apr. 2007, pp. 35–44

[15] Beckhoff, C.; Koch, D.; Torresen, J.: “Go Ahead: A Partial Reconfiguration Framework,” Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium , pp.37-44, April 29 2012-May 1 2012

[16] Bolchini, C.; Miele, A; Santambrogio, M.D.: “TMR and Partial Dynamic Reconfiguration to mitigate SEU faults in FPGAs,” Defect and Fault-Tolerance in VLSI Systems, 2007. DFT '07. 22nd IEEE International Symposium on , vol., no., pp.87,95, 26-28 Sept. 2007

[17] Brelet, P.; Grasset, A.; Bonnot, P.; Ieromnimon, F.; Kritharidis, D.; Voros, N.S., “System Level Design for Embedded Reconfigurable Systems Using MORPHEUS Platform,” VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium on , vol., no., pp.500,505, 5-7 July 2010

[18] Chapman, K.; Jones, L.: “SEU strategies for Virtex-5 devices,” Xilinx Inc., XAPP864, [online]

[19] Command Line Tools User Guide (ug 628), [online], [2013-10-02]

[20] Correcting Single-Event Upset Through Virtex Partial Configuration (xapp 216), [online] [2000-6-1]

[21] Dutton, B. F.; Stroud, C. E.: “Soft core embedded processor based built-in self-test of FPGAs,” Symp. on Defect and Fault-Tol. in VLSI Systems, 2009, pp. 29–37

[22] Emmert, J.; Stroud, C.; Abramovici, M.: “Online Fault Tolerance for FPGA Logic Blocks,” IEEE Trans. VLSI Systems, vol. 15, no. 2, pp. 216–226, 2007.

[23] Ferrandi, F.; Morandi, M.; Novati, M.; Santambrogio, M.D.; Sciuto, D., “Dynamic Reconfiguration: Core Relocation via Partial Bitstreams Filtering with Minimal Overhead,” System-on-Chip, 2006. International Symposium on , vol., no., pp.1,4, 13-16 Nov. 2006

[24] Fons, F.: “Trigonometric Computing Embedded in a Dynamically Reconfigurable CORDIC System-on-Chip,” Reconfigurable Computing: Architectures and Applications, Lecture Notes in Computer Science, Vol. 3985, pp. 122-127, ISSN 0302-9743, Springer,2006

[25] Fons, M.: “Hardware-Software Co-design of an Automatic Fingerprint Acquisition System,” IEEE International Symposium on Industrial Electronics, ISIE 2005 Conference Proceedings, pp. 1123-1128, Dubrovnik, Croatia, June 2005.

[26] Garcia, R.; Gordon-Ross, A.; George, A.D.: “Exploiting Partially Reconfigurable FPGAs for Situation-Based Reconfiguration in Wireless Sensor Networks,” in 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines. IEEE, 2009, pp. 243-246

[27] Gohringer, D.; Becker, J.: “High performance reconfigurable multi-processor-based computing on FPGAs,” Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on , vol., no., pp.1,4, 19-23 April 2010

[28] Hansen, S.G.; Koch, D.; Torresen, J.: “High Speed Partial Run-Time Reconfiguration Using Enhanced ICAP Hard Macro,” Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on , vol., no., pp.174,180, 16-20 May 2011

[29] He Wei; Wang Yueke; Xing Kefei; Chen Li: “SEU readback interval strategy of SRAM-based FPGA for space application,”Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on , vol.4, no., pp.238,241, 10-12 June 2011

[30] Horta, E.; Lockwood, J. W.: “Parbit: A tool to transform bitfiles toimplement partial reconfiguration of field programmable gate arrays (FPGAS),” Dept. Comput. Sci., Washington Univ., Saint Louis, MO,Tech. Rep. WUCS-01-13, 2001

[31] Hubner, M.; Schuck, C.; Kuhnle, M.; Becker, J.: “New 2-dimensional partial dynamic reconfiguration techniques for real-time adaptive microelectronic circuits,” in Proc. IEEE Comput. Soc. Annu. Symp. Emerging VLSI Technol. Archit., 2006, pp. 97–97

[32] Hubner, M.; Tradowsky, C.; Gohringer, D.; Braun, L.; Thoma, F.; Henkel, J.; Becker, J.: “Dynamic Processor Reconfiguration,” Reconfigurable Computing and FPGAs (ReConFig), 2011 Internationa

[33] Ichinomiya, Y.; Amagasaki, M.; Iida, M.; Kuga, M.; Sueyoshi, T.: “A Bitstream Relocation Technique to Improve Flexibility of Partial Reconfiguration,” Algorithms and Architectures for Parallel Processing, 2012 12th International Conference, ICA3PP 2012, vol., no., pp.139/152,Settember 4-7 2012

[34] Ichinomiya, Y.; Tanoue, S.; Amagasaki, M.; Iida, M.; Kuga, M.; Sueyoshi, T.: “Improving the robustness of a softcore processor against seus by using tmr and partial reconfiguration,” in Proceedings of the 2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, ser. FCCM ’10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 47–54

[35] Kalte, H.; Lee, G.; Porrmann, M.; Ruckert, U.: “REPLICA: A Bitstream Manipulation Filter for Module Relocation in Partial Reconfigurable Systems,” Parallel and Distributed Processing Symposium, Proceedings 19th IEEE International, pp. 151b, 04-08 April 2005

[36] Kalte, H.; Porrmann, M.: “REPLICA2Pro: Task Relocation by Bitstream Manipulation in VIRTEX-II/Pro FPGAs,” In:Proceedings of the 3rd Conference on Computing Frontiers.; 2006: 403–412

[37] Khan, F.; Ghiasi, S.; Chen-Nee Chuah: “A Dynamically Reconfigurable System for Closed-Loop Measurements of Network Traffic," Computers, IEEE Transactions on , vol.63, no.2, pp.263,275, Feb. 2014

[38] Krasteva, Y.E.; de la Torre, E.; Riesgo, T.; Joly, D.: “Virtex II FPGA Bitstream Manipulation Application to Reconfiguration Control Systems,” Field Programmable Logic and Applications, 2006. FPL '06. International Conference on , vol., no., pp.1-4, 28-30 Aug. 2006

[39] Llamocca, D.; Pattichis, M.; Alonzo, G.: “Partial Reconfigurable FIR Filtering System Using Distributed Arithmetic,” International Journal of Reconfigurable Computing Volume 2010 (2010)

[40] Miculka, L.; Straka, M.; Kotasek, Z.: “Methodology for Fault Tolerant System Design Based on FPGA into Limited Redundant Area,” Digital System Design (DSD), 2013 Euromicro Conference on , vol., no., pp.227,234, 4-6 Sept. 2013

[41] Milton, D.; Dhingra, S.; Stroud, C. E.: “Embedded processor based built-in self-test and diagnosis of logic and memory resources in FPGAs,” in Int’l Conf. on Embedded Systems and Applications (ESA), 2006, pp. 87–93

[42] Pilotto, C.; Azambuja, J. R.; Kastensmidt, F. L.: “Synchronizing triple modular redundant designs in dynamic partial reconfiguration applications,” in SBCCI ’08: Proceedings of the 21st annual symposium on Integrated circuits and system design. New York, NY, USA: ACM, 2008, pp. 199–204.

[43] Raaijmakers, S. W. S.: “Run-time partial reconfiguration for removal, placement and routing on the Virtex-ii pro,” in Proc. Int. Conf. Field Program. Logic Appl., Aug. 2007, pp. 679–683

[44] Rana, V.; Santambrogio, M.; Sciuto, D.; Kettelhoit, B.; Koester, M.; Porrmann, M.; Ruckert, U.: “Partial Dynamic Reconfiguration in a Multi-FPGA Clustered Architecture Based on Linux,” Parallel and Distributed Processing Symposium, 2007. IPDPS 2007 IEEE International, pp.1,8, 26-30 March 2007

[45] Sari, A.; Psarakis, M.; Gizopoulos, D.: “Combining checkpointing and scrubbing in FPGA-based real-time systems,” VLSI Test Symposium (VTS), 2013 IEEE 31st , pp.1,6, April 29 2013-May 2 2013

[46] Sohanghpurwala, A.A.; Athanas, P.; Frangieh, T.; Wood, A.: “OpenPR: An Open-Source Partial-Reconfiguration Toolkit for Xilinx FPGAs,” Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium, pp.228,235, 16-20 May 2011

[47] Straka, M.; Kastil, J.; Kotasek, Z.: “Fault Tolerant Structure for SRAM-Based FPGA via Partial Dynamic Reconfiguration,” Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on , vol., no., pp.365,372, 1-3 Sept. 2010

[48] Tanoue, S.; Ishida, T.; Ichinomiya, Y.; Amagasaki, M.; Kuga, M.; Sueyoshi, T.: “A novel states recovery technique for the TMR softcore processor,” Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on , vol., no., pp.543,546, Aug. 31 2009-Sept. 2 2009

[49] Verma, V.; Dutt, S.; Suthar, V.: “Efficient On-line Testing of FPGAs with Provable Diagnosabilities,” in Design Automation Conference (DAC), 2004, pp. 498–503.

[50] Vipin, K.; Fahmy, S.A, “A high speed open source controller for FPGA Partial Reconfiguration,” Field-Programmable Technology (FPT), 2012 International Conference on , vol., no., pp.61,66, 10-12 Dec. 2012

[51] Vipin, K.; Fahmy, S.A, “ZyCAP: Efficient Partial Reconfiguration Management on the Xilinx Zynq,” Embedded Systems Letters, IEEE , vol.6, no.3, pp.41,44, Sept. 2014

[52] Virtex FPGA Series Configuration and Readback (xapp 138), [online], [2005-3-11]

[53] Virtex-5 FPGA Configuration User Guide (ug 191), [online], [2012-10-19]

[54] Virtex-6 Family Overview (DS150), [online], [2012-1-19]

[55] Virtex-6 FPGA Clocking Resources (ug 362), [online], [2013-5-9]

[56] Virtex-6 FPGA DSP48E1 Slice (ug 369), [online], [2011-2-14]

[57] Virtex-6 FPGA Memory Resources (ug 363), [online], [2014-2-5]

[58] Virtex-6 FPGA SelectIO Resources (ug 361), [online], [2013-6-21]

[59] Xilinx Constraints Guide (ug 625), [online], [2012-1-18]

[60] Xilinx Inc.: Early Access Partial Reconfiguration User Guide [online] [2009-12-10]

[61] Xilinx Inc.:7 Series FPGAs Overview (ds180), [online], [2012-5-5]

[62] Xilinx Inc.:Partial Reconfiguration User Guide (ug702), [online], [2010-10-5]

[63] Xilinx PicoBlaze 8-bit Embedded Microcontroller User Giude (ug 129), [online], [2011-6-22]

[64] Xilinx Virtrx-6 FPGA Configurable Logic Block (ug 364), [online], [2012-2-3]

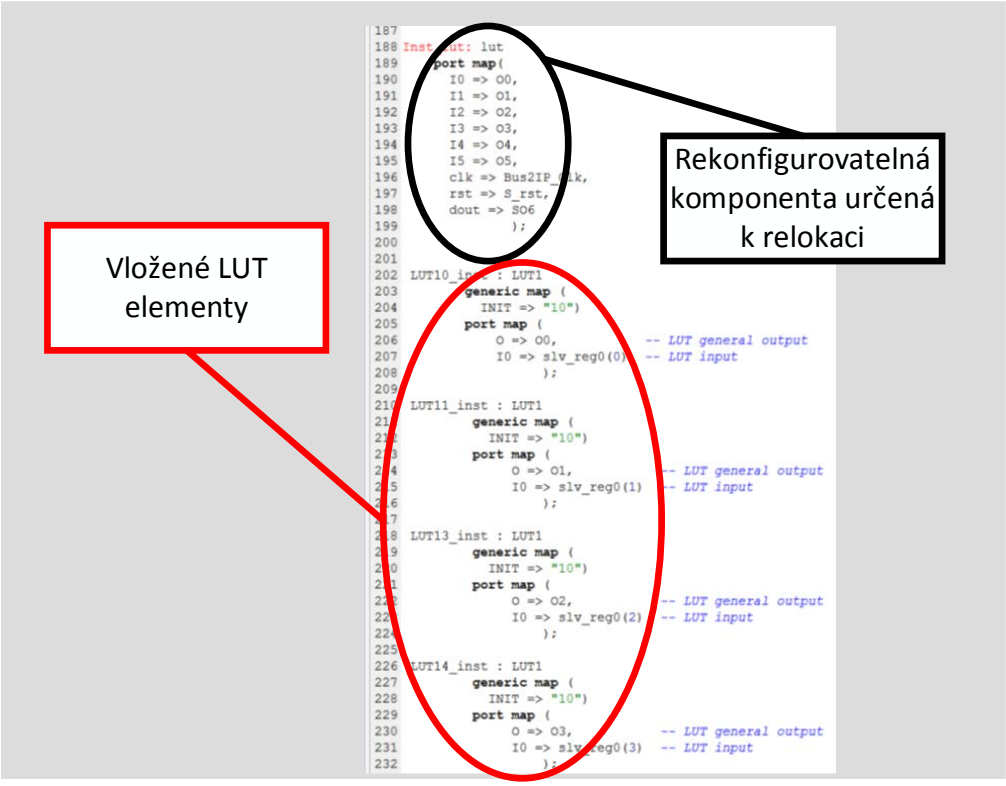
[65] Xilinx Virtrx-6 FPGA Configuration (ug 360), [online], [2012-9-11]

Příloha A – VHDL návrh s přidanými LUT elementy

Za účelem podpory techniky relokace částečných konfiguračních souborů je třeba hardwarový modul, který má být relokován, propojit se statickou částí systému pomocí oddělovacího makra. Naneštěstí toto rozhraní není stávajícími návrhovými prostředky firmy Xilinx podporováno. Z tohoto důvodu byla v rámci této práce vyvinuta metoda umožňující přeměnu aktuálně používaného signálového přemostění, které je založené na použití oddělovacích elementů, na technologicky starší, ale relokaci podporující oddělovací makro.

Tato metoda spočívá v tom, že ke každému oddělovacímu elementu je přidán další LUT element. Tyto dva elementy (vzájemně propojené) spolu tvoří výše zmiňované oddělovací makro (kapitola 4.1.1).

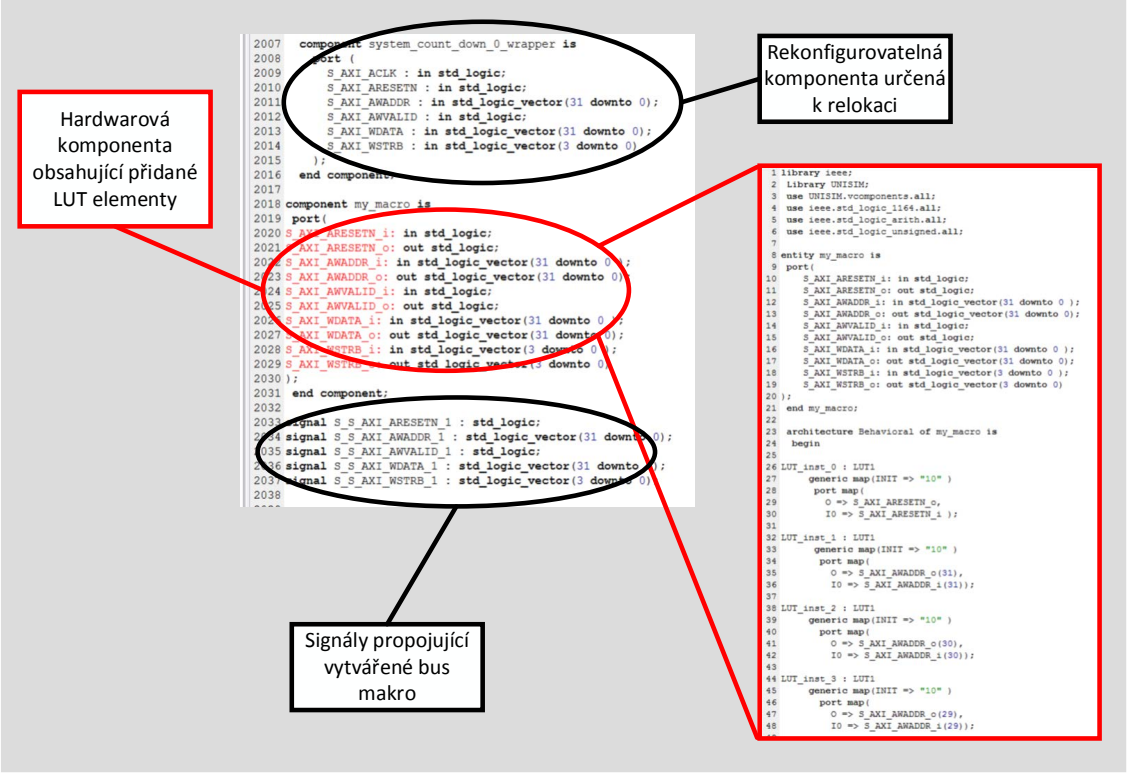
Přidání LUT elementů může být provedeno přímo ve VHDL návrhu, přičemž jednotlivé LUT elementy jsou umístěny v hierarchicky nadřazené části systému, než je samotná komponenta (tj. přidané LUT elementy jsou s rekonfigurovatelnou komponentou na stejné úrovni).



Obr. 58: Rekonfigurovatelná komponenta s přidanými LUT elementy

Na Obr. 58 je graficky vyobrazen fragment VHDL kódu, kde je vidět hardwarový modul určený k relokaci a několik přidaných LUT elementů. Tyto elementy jsou přidávány jako komponenta LUT1. Při větším množství vstupních a výstupních pinů může tento přístup způsobit nepřehlednost té části návrhu, do které jsou tyto elementy přidány.

Další možností je vytvoření samostatné komponenty, ve které jsou LUT elementy umístěny. Tato komponenta je opět přidána v hierarchii návrhu na stejnou úroveň s hardwarovým modulem určeným k relokaci. Tento princip je naznačen na Obr. 59.



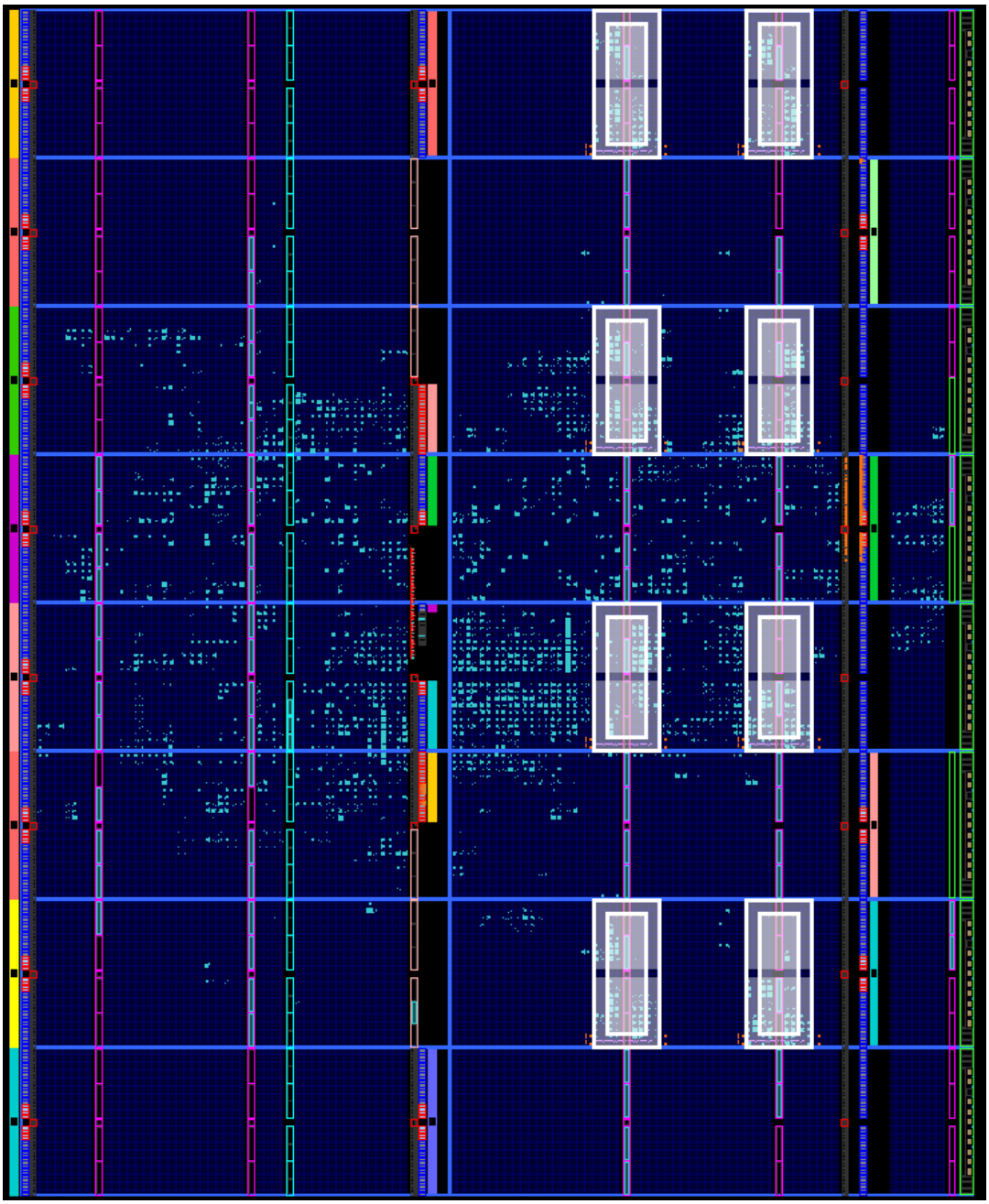
Obr. 59: Přidané LUT elementy umístěné v samostatné hardwarové komponentě

Popsaný přístup je výhodný v tom, že přidané LUT elementy se nacházejí mimo hlavní část VHDL návrhu, a tím pádem zasahují minimálně do jeho složitosti. Takto vytvořenou komponentu je třeba standardně deklarovat (viz Obr. 59) a poté je třeba propojit její vstupy a výstupy s komponentou určenou k relokaci (viz Obr. 60). K tomuto propojení je zapotřebí vytvořit odpovídající počet signálů. Přidávaným LUT elementem jsou opatřeny všechny vstupy a výstupy funkčního modulu kromě vstupu přivádějícího hodinový signál. Hodinový signál je rozváděn po FPGA pomocí vodičů k tomu určených a není tedy osazen ani oddělovacím elementem, ani oddělovacím makrem, aby nedocházelo ke zbytečnému zpoždění tohoto signálu.

```
4335
4336 count_down_0 : system_count_down_0_wrapper
4337
4338  port map (
4339     S_AXI_ACLK => pgassign6(5),
4340     S_AXI_ARESETN => S_S_AXI_ARESETN_1,
4341     S_AXI_AWADDR => S_S_AXI_AWADDR_1,
4342     S_AXI_AWVALID => S_S_AXI_AWVALID_1,
4343     S_AXI_WDATA => S_S_AXI_WDATA_1,
4344     S_AXI_WSTRB => S_S_AXI_WSTRB_1
4345 );
4346
4347 bridge1: my_macro
4348  port map (
4349     S_AXI_ARESETN_i => axi4lite_0_M_ARESETN(4),
4350     S_AXI_AWADDR_i => axi4lite_0_M_AWADDR(159 downto 128),
4351     S_AXI_AWVALID_i => axi4lite_0_M_AWVALID(4),
4352     S_AXI_WDATA_i => axi4lite_0_M_WDATA(159 downto 128),
4353     S_AXI_WSTRB_i => axi4lite_0_M_WSTRB(19 downto 16),
4354     S_AXI_ARESETN_o => S_S_AXI_ARESETN_1,
4355     S_AXI_AWADDR_o => S_S_AXI_AWADDR_1,
4356     S_AXI_AWVALID_o => S_S_AXI_AWVALID_1,
4357     S_AXI_WDATA_o => S_S_AXI_WDATA_1,
4358     S_AXI_WSTRB_o => S_S_AXI_WSTRB_1
4359             );
4360
```

Obr. 60: Fragment VHDL kódu s propojením rekonfigurovatelné komponenty s komponentou obsahující přidané LUT elementy

Příloha B – Testovací systém s procesory PicoBlaze

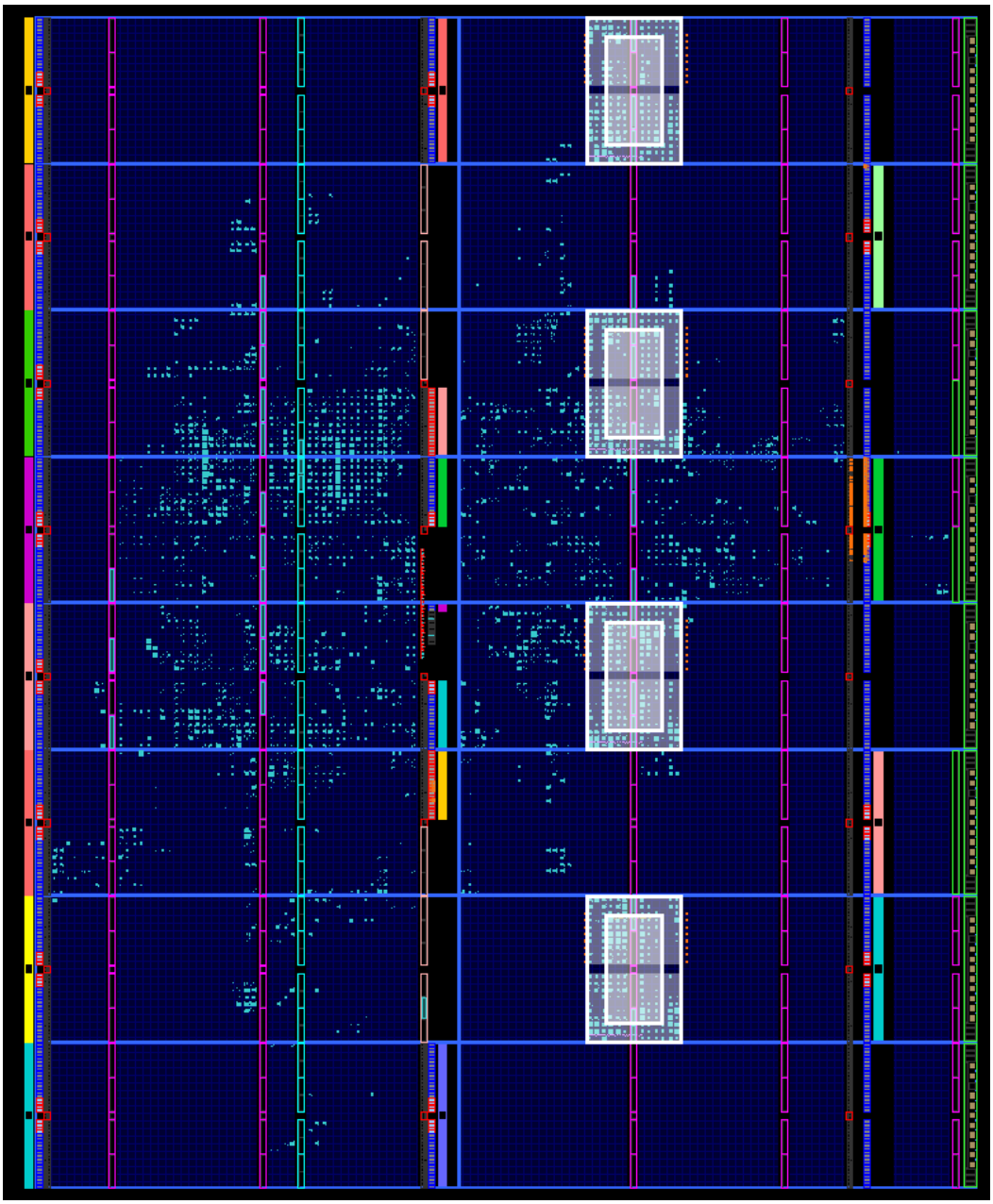


Obr. 61: Uspořádání testovacího systému s procesory PicoBlaze – výřez z grafického rozhraní softwaru PlanAhead

Tab. 14: Využití logických zdrojů v jedné rekonfigurovatelné oblasti s procesorem PicoBlaze

Typ logického prvku	Dostupné množství	Použité množství	Vytíženost [%]
LUT	640	163	26
FD_LD	640	84	14
SLICEL	120	27	23
SLICEM	40	14	35
BRAM	4	1	25

Příloha C – Modifikovaný testovací systém s procesory PicoBlaze



Obr. 62: Uspořádání testovacího systému se čtyřnásobnými procesory PicoBlaze – výřez z grafického rozhraní softwaru PlanAhead

Tab. 15: Využití logických zdrojů v jedné rekonfigurovatelné oblasti se čtyřnásobným procesorem PicoBlaze

Typ logického prvku	Dostupné množství	Použité množství	Vytíženost [%]
LUT	960	652	68
FD_LD	960	336	35
SLICEL	180	116	65
SLICEM	60	48	80
BRAM	4	3	80